Traffic Management for Next Generation Transport Networks

Yu, Hao

Publication date:
2011

Document Version
Publisher's PDF, also known as Version of record

Traffic Management for Next Generation Transport Networks

by

Hao Yu

March, 2011

Networks Technology & Service Platforms
in the
Department of Photonics Engineering
of the
TECHNICAL UNIVERSITY of DENMARK
KGS. LYNGBY
DENMARK

To my parents.

Abstract

Video services are believed to be prevalent in the next generation transport networks. The popularity of these bandwidth-intensive services, such as Internet Protocol Television (IPTV), online gaming, and Video-on-Demand (VoD), are currently driving the network service providers to upgrade their network capacities. However, in order to provide more advanced video services than simply porting the traditional television services to the network, the service provider needs to do more than just augment the network capacity. Advanced traffic management capability is one of the relevant abilities required by the next generation transport network to provide Quality-of-Service (QoS) guaranteed video services.

Augmenting network capacity and upgrading network nodes indicate long deployment period, replacement of equipment and thus significant cost to the network service providers. This challenge may slacken the steps of some network operators towards providing IPTV services. In this dissertation, the topology-based hierarchical scheduling scheme is proposed to tackle the problem addressed. The scheme simplifies the deployment process by placing an intelligent switch with centralized traffic management functions at the edge of the network, scheduling traffic on behalf of the other nodes. The topology-based hierarchical scheduling scheme is able to provide outstanding flow isolation due to its centralized scheduling ability, which is essential for providing IPTV services.

In order to reduce the required bandwidth, multicast is favored for providing IPTV services. Currently, transport networks lack sufficient multicast abilities. With the increase of the network capacity, it is challenging to build a multicast-enabled switch for the transport network, because, from the traffic management's perspective, the multicast scheduling algorithm and the switch architecture should be able

to scale in switch size and link speed. The Multi-Level Round-Robin Multicast Scheduling (MLRRMS) algorithm is proposed for the Input Queuing (IQ) multicast architecture in this dissertation. The algorithm is demonstrated a low implementation and computing complexity, and high performances in terms of delay and throughput. This contribution makes it possible to provide QoS control in a very high-speed switch, such as 100 Gbit/s Ethernet switch.

In addition to the multicast scheduling algorithm, the switch fabric, which is the core of the switching system, should also be able to scale and deliver excellent QoS performances. One challenge is to solve the Out-Of-Sequence (OOS) problem of the multicast cells in the three-stage Clos-network, a type of multistage switch fabrics with a larger scalability than single-stage switch fabrics. In this dissertation, two cell dispatching schemes are proposed for the Space-Memory-Memory (SMM) Clos architecture, which are the Multicast Flow-based DSRR (MF-DSRR) and the Multicast Flow-based Round-Robin (MFRR). Both schemes are capable of reducing the OOS problem, and thus decrease the reassembly delay and buffer size. This improvement is of great significance for the multicast switching service, which is foreseen to be extensively used in the next generation transport network.

To sum up, this dissertation discusses the traffic management for the next generation transport network, and proposes novel scheduling algorithms to solve some of the challenges currently encountered by both the academia and the industry. The covered topics in this dissertation are related to the two projects: *High quality IP network for IPTV and VoIP* (HIPT) and *The Road to 100 Gigabit Ethernet* (100GE), which are detailed in the dissertation.

Résumé

Videotjenester forventes at blive fremherskende i den næste generation transportnetvæk. Populariteten båndbreddekrævende tjenester, såsom Internet TV, online gaming, og Video-on-Demand (VoD), stiller krav til internet-udbydere om at opgradere deres netværk til den krævede kapacitet. For at tilbyde mere avancerede video-tjenester end blot traditionel tv, er der behov for at yderligere tiltag end simpel forøgelse af båndbredden. En af de nødvendige tiltag er avanceret trafikstyring af fremtidens netværk, således at der sikre høj kvalitet (QoS) og garanteret båndbredde til de forventede videotjenester.

Forøgelse af netværkskapacitet og opgradering af knudepunkter har en lang tidshorisont og kræver betydelige omkostninger for for netværksudbyderne. Disse udfordringer kan begrænse udbydernes lyst til at levere Internet Protocol Television (IPTV)-tjenester. I denne afhandling foreslås en topologi-baseret hierarkisk skeduleringsmetode med henblik på løse dette problem. Metoden forenkler implementeringen ved at placere en intelligent switch med centraliseret trafikstyringfunktionalitet på kanten af nettet, hvor den styrer trafikken på vegne af de øvrige knudepunkter. Denne topologi-baserede hierarkiske skeduleringsmetode er i stand til at levere fremragende opdeling af trafik-flows grundet dens centrale skeduleingsevne, som er afgørende for at tilbyde IPTV-tjenester.

For at reducere den krævede båndbredde ønskes at benytte multicast til at levere IPTV-tjenester. Idag har de tilgængelige transportnetværk dog ikke tilstrækkelig understøttelse af multicast, og med forøgelsen af netværkskapaciteten er det krævende at opbygge en multicast-understøttet switch til transport netværket, da det fra et trafikstyringsperspektiv er nødvendigt at multicast skeduleringsalgoritmen og switchens arkitektur kan skaleres i forhold til størrelse og hastigheden af dens forbindelser.

Multi-Level Round-Robin Multicast Scheduling (MLRRMS) algoritmen er foreslået som Input Queuing (IQ) multicast arkitektur i denne afhandling. Algoritmen er påvist at have en beskeden implementerings- og databehandlingskompleksitet smatidig med med høj ydeevne i form af forsinkelse og båndbredde. Dette bidrag gør det muligt at yde Quality-of-Service (QoS) styring i en ekstrem høj-hastighedsswitche, som f.eks. 100 Gbit/s Ethernet switche.

Udover multicast skeduleringsalgoritmen, skal switch bagplanet, som er kernen i switch-systemet, kunne skalere og levere fremragende QoS resultater. En af udfordringerne er at løse Out-Of-Sequence (OOS) problemet med multicast celler i tre-trins Clos-netværk, som er en slags flertrins switch bagplan højere skalerbarhed end en et-trins switch-bagplan. I denne afhandling foreslås to løsninger til afsendelse af celler i Space-Memory-Memory (SMM) Clos arkitekturer. Disse er Multicast Flow-based DSRR (MF-DSRR) og Multicast Flow-based Round-Robin (MFRR). Begge løsninger er i stand til at reducere OOS problemet og dermed mindske buffer-dybde og forsinkelse i forbindelse med genetableringen af data-pakken. Denne forbedring er af stor betydning for multicast tjenester, som forventes at blive flittigt brugt i fremtidens transport-netværk.

Opsummerende diskuterer denne afhandling trafikstyringen for fremtidens transportnetværk og foreslår nye skeduleringsalgoritmer til at løse nogle af de udfordringer, som idag findes i både den akademiske verden og i industrien. De dækkede emner i denne afhandling er relateret til de to projekter: Høj kvalitet IP netværk til IPTV og VoIP (HIPT) og The Road to 100 Gigabit Ethernet (100GE), som er beskrevet i afhandlingen.

Acknowledgement

With their continued guidance, support and inspiration, I would like to thank my supervisor Professor Lars Dittmann, Dr. Michael S. Berger, and Dr. Sarah Ruepp. It has been an arduous yet pleasant journey to pursue my Ph.D during the stay at DTU. I really appreciate the encouragement that Professor Lars Dittman gave me back in 2008 before this journey. Without him, it would be impossible for me to experience the beauty of the pursuit of my Ph.D.

I am grateful to Dr. Michael S. Berger for all the discussions and help on the three topics in this thesis, and the freedom of the research environment that you have given to me and your other students. Your inspiration has led me to many of my accomplishments, and I deeply thank you for all the encouragements on numerous situations over the years.

I would like to give my appreciation to Dr. Sarah Ruepp for all the kindly help and support on the work of the projects. Your preciseness has left me a great impression and it is enjoyable and agreeable to work with you.

A special thank should be expressed to Dr. Ying Yan. The close collaboration with you on the HIPT project gave me countless experiences which greatly helped me in the first year of my Ph.D study.

Thanks to Dr. Villy Bæk Iversen for the inspiring discussions about the analytical analysis on the multi-level round-robin multicast scheduling algorithm. Your solid traffic engineering and probability skills have helped and inspired me greatly.

Thanks to all the my other colleagues in the Network Technology and Service Platform group: Dr. Henrik Wessing, Dr. José Soler, Dr. Lars Staalhagen, Dr. Anna Vaseliva Manolova, Rong Fu, Jiang Zhang,

Lukasz Brewka, Ana Rossello, Anders Rasmussen, Anna Zakrzewska, Jiayuan Wang, Thang Tien Pham, and Brian Sørensen. You have made this group a pleasant place to work in and it is my great pleasure to spend these years with you.

Thanks again to Dr. Sarah Ruepp, Dr. Henrik Wessing, Dr. José Soler, Jiang Zhang, and Dr. Ying Yan for proofreading and commenting on this thesis.

Last, and absolutely most, I would like to express my gratitude to my parents. I dedicate this thesis to you, for your understanding and support that have given me strength over the years.

Ph.D Publications

The following publications have been made throughout this Ph.D project.

Publications on the topic: IPTV traffic management in Carrier Ethernet transport networks

[1] H. Yu, Y. Yan, and M. S. Berger, "IPTV traffic management in Carrier Ethernet transport networks," in *OPNETWORK 2008*, 2008

[2] H. Yu, Y. Yan, and M. S. Berger, "IPTV traffic management using topology-based hierarchical scheduling in Carrier Ethernet transport networks," in *International Conference on Communications and Networking in China (ChinaCom)*, pp. 1–5, 2009

[3] H. Yu, Y. Yan, and M. S. Berger, "Topology-based hierarchical scheduling using deficit round robin: Flow protection and isolation for triple play service," in *First International Conference on Future Information Networks*, pp. 269–274, 2009

[4] A. Rasmussen, J. Zhang, H. Yu, R. Fu, S. Ruepp, H. Wessing, and M. S. Berger, "Towards 100 gigabit Carrier Ethernet transport networks," *WSEAS Transactions on Communications*, vol. 9, pp. 153–164, 2010

[5] H. Wessing, M. S. Berger, H. Yu, A. Rasmussen, L. Brewka, and S. Ruepp, "Evaluation of network failure induced IPTV degradation

in metro networks," *Recent Advances in Circuits, Systems, Signal and Telecommunications*, pp. 135–139, 2010

[6] H. Wessing, M. S. Berger, H. M. Gestssson, H. Yu, A. Rasmussen, L. Brewka, and S. Ruepp, "Evaluation of restoration mechanisms for future services using Carrier Ethernet," *WSEAS Transactions on Communications*, vol. 9, pp. 322–331, 2010

Publications on the topic: Multicast scheduling for input-queued high-speed switches

[1] H. Yu, S. Ruepp, and M. S. Berger, "A novel round-robin based multicast scheduling algorithm for 100 gigabit ethernet switches," in *29th IEEE International Conference on Computer Communications (INFOCOM) Workshops*, pp. 1–2, 2010

[2] H. Yu, S. Ruepp, and M. S. Berger, "Round-robin based multicast scheduling algorithm for input-queued high-speed Ethernet switches," in *OPNETWORK 2010*, 2010

[3] H. Yu, S. Ruepp, and M. S. Berger, "Enhanced fifo based round-robin multicast scheduling algorithm for input-queued switches," *IET Communications*, vol. 5, pp. 1163–1171, 2011

[4] H. Yu, S. Ruepp, and M. S. Berger, "Multi-level round-robin multicast scheduling with look-ahead mechanism," in *IEEE International Conference on Communications*, 2011

Publications on the topic: Out-of-sequence prevention for multicast Clos-network

[1] H. Yu, S. Ruepp, and M. S. Berger, "Out-of-sequence prevention for multicast input-queuing space-memory-memory Clos-network," *IEEE Communications Letters*, 2011

[2] H. Yu, S. Ruepp, and M. S. Berger, "Out-of-sequence preventative cell dispatching for multicast input-queued space-memory-memory Clos-network," in *12th IEEE International Conference on High Performance Switching and Routing*, 2011

Publications on the topic: Integrated control platform design in converged optical and wireless networks

[1] Y. Yan, H. Yu, and L. Dittmann, "Wireless channel condition aware scheduling algorithm for hybrid optical/wireless networks," in *3rd. International Conference on Access Networks*, pp. 397–409, 2008

[2] Y. Yan, H. Yu, H. Wang, and L. Dittmann, "Integration of EPON and WiMAX networks: Uplink scheduler design," in *SPIE Symposium on Asia Pacific Optical Communications*, 2008

[3] Y. Yan, H. Yu, H. Wessing, and L. Dittmann, "Integrated resource management for hybrid optical wireless (how) networks," in *International Conference on Communications and Networking in China (ChinaCom)*, pp. 1–5, 2009

[4] Y. Yan, H. Yu, H. Wessing, and L. Dittmann, "Enhanced signaling scheme with admission control in the hybrid optical wireless (HOW) networks," in *28th IEEE International Conference on Computer Communications (INFOCOM) Workshops*, pp. 1–6, 2009

[5] Y. Yan, H. Yu, H. Wessing, and L. Dittmann, "Integrated resource management framework in hybrid optical wireless networks," *IET Optoelectronics Special Issue on Next Generation Optical Access*, vol. 4, pp. 267–279, 2010

This dissertation only includes work for the topic on (1) IPTV traffic management in Carrier Ethernet transport networks, (2) Multicast scheduling for input-queued high-speed switches, and (3) Out-of-sequence prevention for multicast Clos-network.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

"The best way to predict the future is to invent it."

For the last two decades, with the rapid development of telecommunication technologies, the network bandwidth capacity has increased significantly, both in the access and the transport areas. This has led to a boom of network applications that require broadband access and high network capacity, such as High Definition (HD) Video-on-Demand (VoD), video sharing, videoconferencing, and online gaming, so on and so forth. At the same time, the recent invention and development of such applications, which require high network bandwidth, have speeded up the growth of the network capacity and have generated more demands on the transmission speed. It is foreseeable that this trend will continue in the near future, and network operators will keep upgrading the capacity of their networks. However, the pursuit of increasing the network capacity alone cannot provide customers with excellent experiences of the applications, without proper traffic management functions to classify, schedule, and monitor the enormous amount of network traffic.

Quality-of-Service (QoS) has been an issue of relevance for many years, ever since the development of Internet applications are not limited to best effort data applications, such as plain web browsing. To guarantee the QoS is of great importance to network operators because it is the foundation of providing many applications, including HD-VoD, Internet Protocol Television (IPTV), and Voice-over-IP (VoIP). Without the ability to provide QoS guarantees, the IPTV traffic, for instance,

can be suddenly delayed or the voice call can be unexpectedly dropped due to an increase in network traffic load, which undoubtedly affects the user experience and therefore the popularity of the application. Traffic management aims to schedule different traffic, avoid congestions, and allocate bandwidth, in order to provide a fine-grained QoS ability to the network.

Developed in the 70's, Ethernet is a frame-based networking technology standardized in IEEE 802.3, and is originally for Local Area Networks (LANs). As shown in Table 1.1, Ethernet has been evolving from 10 Mbit/s to today's 100Gbit/s. In addition to higher bandwidth, the evolution contains improved Media Access Control (MAC) schemes and physical medium changes as well, which are out of the scope of this dissertation.

	Year	**Standards**	**Bit Rate**
Ethernet	1985	IEEE 802.3a	10 Mbit/s
Fast Ethernet	1995	IEEE 802.3u	100 Mbit/s
Gigabit Ethernet	1999	IEEE 802.3ab IEEE 802.3ah	1000 Mbit/s
10 Gigabit Ethernet	2002	IEEE 802.3ae	10 Gbit/s
40 Gigabit Ethernet	2010	IEEE 802.3ba	40 Gbit/s
100 Gigabit Ethernet	2010	IEEE 802.3bg	100 Gbit/s

Table 1.1: A Brief summary of the evolution of Ethernet.

Ethernet has successively dominated the LAN networks for decades and the evolution of Ethernet has been increasing from several megabits per second to today's 100 gigabits per second, driven by the development of various applications. Since LAN networks are increasingly connected to the Metropolitan Area Network (MAN) over Ethernet interfaces, it drives the operators to provide Ethernet services in their MAN networks. Due to the domination of Ethernet, Carrier Ethernet is defined by the Metro Ethernet Forum (MEF) [18] as an extension to enable telecommunication operators to provide standardized Ethernet services

to the customers, such as E-Line, E-LAN and E-tree services [19, 20]. The transport network is evolving from the legacy technology that provides constant bit rate connection to today's Carrier Ethernet technologies, which is capable of providing flexible bandwidth and services through packet switching. Two main candidates to the Carrier Ethernet technologies are the Provider Backbone Bridge with Traffic Engineering (PBB-TE) defined in IEEE 802.1 Qay [21], and Multi-protocol Label Switching Transport Profile (MPLS-TP) [22], jointly developed by the International Telecommunication Union-Telecommunication Standardization Sector (ITU-T) and the Internet Engineering Task Force (IETF).

It has widely been discussed that, Carrier Ethernet can become the main framework for the next generation transport networks [19, 23–28]. However, advanced traffic management functionalities are integral for the Carrier Ethernet transport network, in order to provide guaranteed QoS to various applications. Thus, this dissertation focuses on the development of effective traffic management mechanisms for switches in the next generation Carrier Ethernet transport network, to suffice various QoS requirements. This implies that, traffic management on the Internet Protocol (IP) layer, IP lookup, IP routing and other Layer 3 (L3) technologies are out of the scope of this dissertation. The term, *switch*, is used throughout this dissertation to indicate either the Layer 2 (L2) switch or the switch engine used by IP routers.

Scheduling technology plays an essential role in the traffic management, because it is the scheduling algorithm that steers the QoS performance of a switch and finally the entire network. As packets arrive at a switch, a packet-based scheduler schedules packets according to the QoS requirements of different packet streams. Before packets being transmitted to the switch fabric, the core of the switch, they are usually segmented into fixed-size pieces, known as cells. The purpose of doing so is to increase the throughput and reduce the scheduling complexity [29]. A cell-based scheduler is responsible for selecting cells to transmit to the switch fabric. As an enormous amount of cells enter into the switch fabric, cells are arranged by a scheduling algorithm on the lower in-switch fabric level of traffic management. Figure 1.1 illustrate the relation between different levels of scheduling.

Given the increased network capacity, it is challenging for both the academia and the industry to give solutions to the traffic management

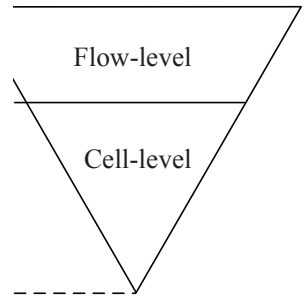


Figure 1.1: Illustration of different levels of traffic scheduling.

of switches in the next generation high-speed transport networks. The issues of scalability, complexity, and performance should be taken into consideration. Hence, the work in this dissertation can be summarized as:

> **The objectives of this dissertation is to develop novel traffic management mechanisms for the next generation transport networks, in order to provide improved traffic QoS guarantee. The work considers the scalability and the complexity as key factors, and aims to improve the QoS performance. Scheduling algorithms on different levels and scales are proposed, and simulations are carried out for evaluation.**

Following the illustration of the relation between scheduling on different levels, the outline of this dissertation is organized as follows:

Chapter 2 presents the motivations of the work of this dissertation within the scopes of two projects *High Quality IP network for VoIP and IPTV* (HIPT) and *The Road to 100 Gigabit Ethernet* (100GE).

Chapter 3 concentrates on the traffic scheduling algorithms for IPTV in the Carrier Ethernet transport networks, aiming to provide an improved end-to-end QoS. This chapter discusses the possibility and the benefits of centralizing intelligent traffic management functions to the edge of the network, and examines different packet-based scheduling algorithms. The topology-based hierarchical scheduling algorithm is proposed to reduce the network deployment cost and at the same time to

provide flow isolation, which is critical to IPTV traffic. The work in this chapter is included in [1–3].

Chapter 4 explores the multicast scheduling algorithms for switches. Taking the scalability and the implementation complexity into account, this chapter proposes a novel cell scheduling algorithm for input-queued multicast switches. The proposed sync mechanism enables the switch to reduce the unnecessary multiple transmissions of a multicast cell, without affecting the output port utilization. The proposed look-ahead mechanism is able to increase the throughput of the input queuing architecture by reducing the head-of-line blocking. The work in this chapter is enclosed in [7–10].

Chapter 5 investigates the multicast scheduling algorithms and cell dispatching schemes inside the switch fabric of the three-stage Clos-network architecture. In order to prevent the out-of-sequence problem, two novel cell dispatching schemes are proposed for the input-queued space-memory-memory Clos-network. Two types of out-of-sequence problems are defined in this chapter in order to evaluate the different cell dispatching schemes. The work in this chapter is included in [11, 12].

Finally, Chapter 6 presents the conclusion of this dissertation and addresses the future research.

Chapter 2

Motivation

In this chapter, the motivation of the dissertation is presented within the scope of two projects.

For many years, telecommunication service providers have been seeking new ways out to balance the declining revenue on the traditional telephone service and broadband access. It is believed that the introduction of Internet Protocol Television (IPTV) service is the next step that can deliver a substantial increase in revenue to the operators. As a response to the increased interest in IPTV, the Danish Advanced Technology Foundation decided to finance a research project entitled *High quality IP network for IPTV and VoIP* (HIPT). The objective of the HIPT project is to enhance the Carrier Ethernet transport network for IPTV applications, by developing technologies that can fulfill the increasing requirements, such as integrating control plane, traffic management, extended surveillance mechanisms and methods for protection, redundancy and resiliency, and at the same time, can reduce the cost of network operation.

Although IP and Layer 3 (L3) have proven useful in addressing the Internet and other best effort data applications, this approach is not well suited to high-bandwidth, critical services, such as IPTV, which cannot tolerate delays in the network in general. The HIPT project intends to investigate whether intelligent Layer 2 (L2) and Layer 1 (L1) networks can be used to alleviate the problems seen in the current IPTV networks. Using Provider Backbone Bridge with Traffic Engineering (PBB-TE) and Multi-protocol Label Switching Transport Profile (MPLS-TP), au-

tonomous network decision making is removed and more traffic engineering is performed in the network. This ensures control over exactly where traffic is being transported in the network with the further ability to monitor individual traffic flows, which is not easy to accomplish in L3 [30]. This approach has the further advantage of being able to support L3. Rather than replacing L3, the solution intends to supplement it by reducing the need for costly nodes supporting services. Thus, instead of deploying a large number of nodes with large complexity, a simpler, yet intelligent, L2/L1 network based on Carrier Ethernet transport technologies can reduce cost and complexity, while enabling independent scaling of IPTV services at L3 with fewer nodes. The Carrier Ethernet based network architecture for IPTV transport is based on a L2 approach with L3 support in the edge routers and L3 awareness in the Digital Subscriber Line Access Multiplexer (DSLAM) as shown in Figure 2.1.

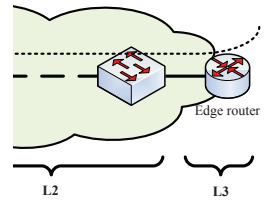


Figure 2.1: HIPT network architecture.

For simplicity, only the DSLAM access is shown, but other types of access technologies can also be applied. IPTV traffic flows are terminated in the Set Top Box (STB) in the home network. The L2 network between the Internet Protocol (IP) DSLAM and the edge router is assumed to be based on either MPLS-TP or PBB-TE. In both cases, the goal is to transport the IPTV traffic with carrier-class quality, and at the same time to reduce the cost by utilizing the Carrier Ethernet technologies. This demands that the L2 Carrier Ethernet network is able to deliver sufficient capacity and traffic management capabilities. The goal of the HIPT project is to develop high-capacity Carrier Ethernet network nodes with advanced traffic management, Operation, Administra-

tion and Maintenance (OAM), and support to guarantee the transport of demanding real-time applications, such as IPTV. Research challenges include the QoS-enabled Carrier Ethernet control plane, OAM for IPTV flow monitoring, resiliency and survivability, and traffic management with end-to-end QoS guarantee.

Given the increasing popularity of IPTV services and other high-bandwidth applications, the Ethernet transmission speed has evolved from 10 Mbit/s to today's 100 Gbit/s. *The Road to 100 Gigabit Ethernet* (100GE) is an ongoing project funded by the Danish Advanced Technology Foundation, aiming at scaling the Ethernet capacity from current 10 Gbit/s to the next generation 100 Gbit/s. The challenges will include interface adaptation, data and control plane processing, high-speed switching, power and printed circuit board design. The goal of achieving the speed of 100 Gbit/s is ambitious and requires improvements in the low-level circuit technology. However, the improvements in the low-level circuit technology will be far from enough to guarantee the switching performance when moving to 100 Gbit/s. Advanced packet processing and switching technologies with a high degree of scalability are indispensable.

At the speed of 100 Gbit/s, the processing time of an Ethernet packet can down to only 5 ns, which is extremely short. The access to external memories for traffic management is a substantial challenge because the speed of memory does not follow the Moore's law that the processing capacity will double around every second year. Therefore a highly scalable and low-complexity scheduling algorithm for switching become significant for the project.

The work in this dissertation includes the IPTV traffic management for Carrier Ethernet transport networks in the HIPT project, and the high-speed multicast scheduling algorithm and cell dispatching algorithms in the 100GE project.

Chapter 3

Topology-based Hierarchical Scheduling

Carrier Ethernet is becoming a favorable transport technology for the Next Generation Network (NGN). The features of cost-efficiency, operation flexibility and high bandwidth have a great attraction to service providers [20,24]. However, to achieve these characteristics, Carrier Ethernet needs to obtain the required provisioning abilities, which guarantee the end-to-end performances of voice, video and data traffic delivered over the network.

Switches with class-based scheduling algorithms schedule traffic based on different QoS classes. Although simple to implement, the class-based scheme lacks the ability to isolate different traffc flows, which belong to the same QoS class, because packets of the same QoS class are stored in the same queue. Any malicously behaving traffic flow can affect the other conforming traffic of the same QoS class, resulting in the vulnerability to traffic attack.

Switches with flow-based scheduling algorithms, on the other hand, are able to protect each traffic flow from being affected by others. This is implemented by further dividing the traffic of the same QoS class into different queues, based on information such as port number, source address, or destination address. However, in order to provide end-to-end QoS guarantees, the network operator needs to upgrade all the switches in the network, which is difficult and costly to implement.

In this chapter, a topology-based hierarchical scheduling scheme is

proposed to provide an alternative solution to provide end-to-end QoS guarantees. The main idea of the topology-based hierarchical scheduling is to map the topology of the connected network into the logical structure of the scheduling system, and to combine several token schedulers according to the topology. The mapping process can be completed through the network management plane or by manual configuration, which is out of the scope of this chapter. Based on the knowledge of the network topology, the scheduler can manage the traffic on behalf of other less advanced nodes in the network, avoiding potential traffic congestion, and providing flow protection and isolation.

Comparisons among the topology-based hierarchical scheduling, the flow-based scheduling, and the class-based scheduling algorithms are carried out under a symmetric binary tree topology. Simulation results show that the topology-based hierarchical scheduling algorithm outperforms the others, in terms of flow protection and isolation from the attack of malicious traffic. This is significant for Internet Protocol Television (IPTV) services in the Carrier Ethernet transport networks.

3.1 Introduction

Ethernet, an incontestable technology that has dominated the Local Area Network (LAN) for decades, is now being developed and extended to become a possible choice for the Metropolitan Area Network (MAN). The pressure from competition and the changing needs of communications and entertainment of residential customers are driving operators to upgrade their networks to be capable of voice, video and data delivery (also known as triple-play services). Most voice, video, and data services used to be provided by separated networks, such as Public Switched Telephone Network (PSTN), the cable television network, and the Internet. The tendency of today is to integrate the services on a single network. In such a network, video broadcasting/multicasting and Video-on-Demand (VoD) services on networks (also known as services) will significantly increase the traffic load. To ensure that the quality of IPTV services is guaranteed without damaging Voice-over-IP (VoIP) services and high-speed Internet access, different QoS requirements of each type of traffic must be ensured by the converged network. Thus, a fine-grained traffic management scheme is demanded.

Each type of service has different QoS requirements [31, 32]. Developing a set of traffic management functions to meet various requirements for each service is an important issue. The network can be subjected to a very heavy traffic load for a certain period. Especially at the edge node, a large amount of incoming traffic compete for the output bandwidth. Hence, it is relevant to discriminate different services and provide guaranteed QoS performance, so that a bandwidth-hungry user does not cause performance degradation to other users in the network.

The Metro Ethernet Forum (MEF) [18] has provided a clear definition of Carrier Ethernet. Based on the description from MEF, Carrier Ethernet is defined as an omnipresent, standardized, carrier-class service and network defined by five attributes that distinguish Carrier Ethernet from LAN based Ethernet:

- Standardized Services
- Scalability
- Reliability
- Quality of Service
- Service Management

To use Ethernet as a transport technology, which requires customer separation and manageability, Provider Backbone Bridge with Traffic Engineering (PBB-TE) [21] and Multi-protocol Label Switching Transport Profile (MPLS-TP) [22] have been developed and proposed as carrier grade Ethernet transport network solutions. PBB-TE is a recent development after several years of work by the Institute of Electrical and Electronics Engineers (IEEE) aiming at improving and enhancing Ethernet technology for the use in carrier networks. PBB-TE reuses current implementations of Virtual Local Area Networks (VLANs) and combines it with the network separation and layering principles of PBB [19, 23]. MPLS-TP, the former Transport MPLS (T-MPLS), is now developed under the cooperation of International Telecommunication Union-Telecommunication Standardization Sector (ITU-T) and Internet Engineering Task Force (IETF). It promises a solution that provides familiar and reliable packet-based technology, i.e. , in a way that is aligned with circuit-based transport networks. Both technologies aim to provide a

connection-oriented packet switching transport network, where traffic is tunneled and delivered to the destinations [26].

As shown in Figure 3.1, Carrier Ethernet contains two separate and independent domains, the control plane and the transport plane. The specification of the control plane implementation is not yet finished in the process of standardization. The main functions of the control plane include, however, QoS mapping, label distribution, Call Admission Control (CAC) [27,28]. In the transport plane, traditional switches should be updated with advanced functionalities in order to provide carrier grade services and to guarantee the QoS performance, especially for real-time traffic such as IPTV.

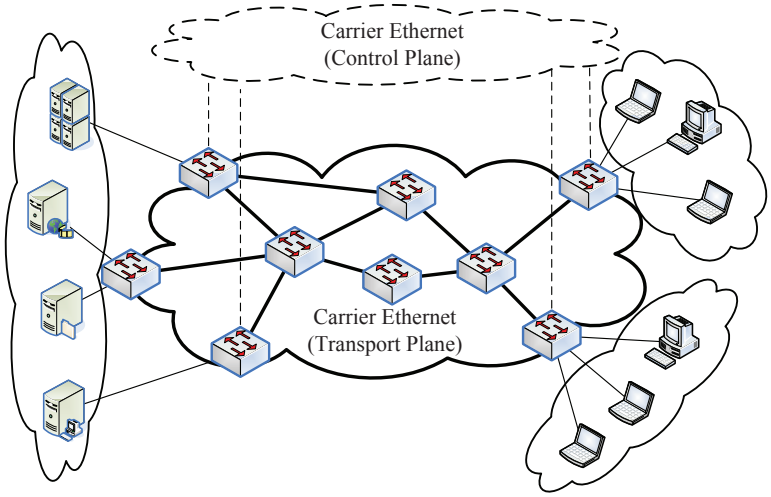


Figure 3.1: The concept of the control plane and transport plane of a Carrier Ethernet network.

From the work of [1], the flow-based scheduling scheme using Deficit Round Robin (DRR) [33] algorithm has been evaluated and has shown to be an appropriate choice for the IPTV service in Carrier Ethernet transport networks. Although the flow-based scheduling scheme is capable of treating traffic flows separately and providing better protection than the class-based scheduling scheme, it requires the network operator to upgrade the entire network with flow-based scheduling nodes, which demand high volumns of buffers. Under economic consideration, net-

work operators consider not only the capability of the network, but also the corresponding cost to deploy and maintain such a network. It has been discussed in [24, 25] that, Carrier Ethernet can greatly reduce the consequences of the complexity associated with the large scale of carrier networks by being a cost-effective replacement for Synchronous Optical Networking (SONET)/Synchronous Digital Hierarchy (SDH) [34].

To keep the preferable features of Carrier Ethernet and to reduce the required deployment period, the topology-based hierarchical scheduling scheme is proposed in this chapter. The term, hierarchical scheduling, has been mentioned and discussed actively in other researchers work. In [35–38], hierarchical scheduling is mainly discussed as an improvement to the traditional DRR for a single network node. There still lacks a hierarchical scheduling scheme that takes the network topology into consideration. Given the detailed topology of the network, where nodes are incapable of flow management, the topology-based hierarchical scheduling node is able to avoid traffic congestion and guarantee QoS requirements. Since the interior nodes of the network may only provide simple forwarding abilities, intelligence can be condensed in the hierarchical scheduling nodes at the edge of the network. By learning the topology of the connected network, the hierarchical scheduler will be able to schedule packet on behalf of other interior nodes.

The remaining parts of this chapter are structured as follows. In Section 3.2, different related scheduling algorithms are compared and the advantages of the DRR scheduling algorithm are explained. In Section 3.3 the system model is presented and the problem is defined. Section 3.4 discusses the benefit of hierarchical scheduling and demonstrate the concept. Section 3.5 presents and analyzes the simulation results. Finally, Section 3.6 concludes the chapter.

3.2 Related Work

Scheduling algorithms are used in a switch design in order to attain the QoS requirement and fairly allocate limited resources among traffic flows. A significant amount of research has contributed to the development of scheduling algorithms [33, 35, 39–52], and they can be mainly divided into two categories: timestamp-based scheduling (also known as sorted-priority scheduling) and frame-based scheduling.

Timestamp-based schedulers maintain a global virtual time to emulate the ideal Generalized Processor Sharing (GPS) [39]. Arriving packets are marked with timestamps which are generated through the virtual machine. The timestamps are used by the scheduler to determine the order of packet departure. This category includes the Weighted Fair Queuing (WFQ) [41], the Worst-case Fair Weighted Fair Queuing (WF^2Q) [42], the Self-Clocked Fair Queuing (SCFQ) [43], and the Start-time Fair Queuing (SFQ) [44]. These timestamp-based schedulers can provide good fairness and low latency. However, a main drawback is that, these methods are not efficient enough due to the complexity involved in computing the system virtual time, and sorting the packets based on the timestamps [53]. The WFQ and the WF^2Q scheduling schemes require $O(N)$ time complexity to complete a scheduling decision, where N denotes the number of active sessions or flows sharing the outgoing link of the switch. The SCFQ approach reduces the time complexity but still holds the $O(\log N)$ bottleneck. Using this kind of schedulers can hinder the scalability of the switching system.

On the other hand, frame-based schedulers serve packets in a round robin manner, i.e. during each round, at least one flow receives a transmission opportunity. This category includes the Deficit Round Robin (DRR) [33], the Elastic Round Robin (ERR) [45], the Carry-Over Round Robin (CORR) [47], and the Mini Round Robin (MRR) [48]. These schedulers do not need to calculate the virtual time, and thus have low time complexities and the design of such frame-based schedulers is simpler compared to timestamp-based schedulers. DRR is one of the early frame-based scheduling algorithms proposed to overcome the unfairness and has a time complexity of $O(1)$, which is much lower than other timestamp-based algorithms. It has been concluded in [33] that the DRR provides near-perfect isolation at low implementation cost and can be combined with other fair queuing algorithms to offer better latency bounds. Low time complexity is a significant factor to a switch design, especially for a high speed link.

Together with advanced buffer management, the DRR algorithm can support sufficient QoS differentiation between flows and can guarantee that any maliciously behaving flow does not affect the QoS performance of other conforming traffic flows. The traditional way of dividing packets is based on their *QoS classes*, as shown in Figure 3.2. Different QoS

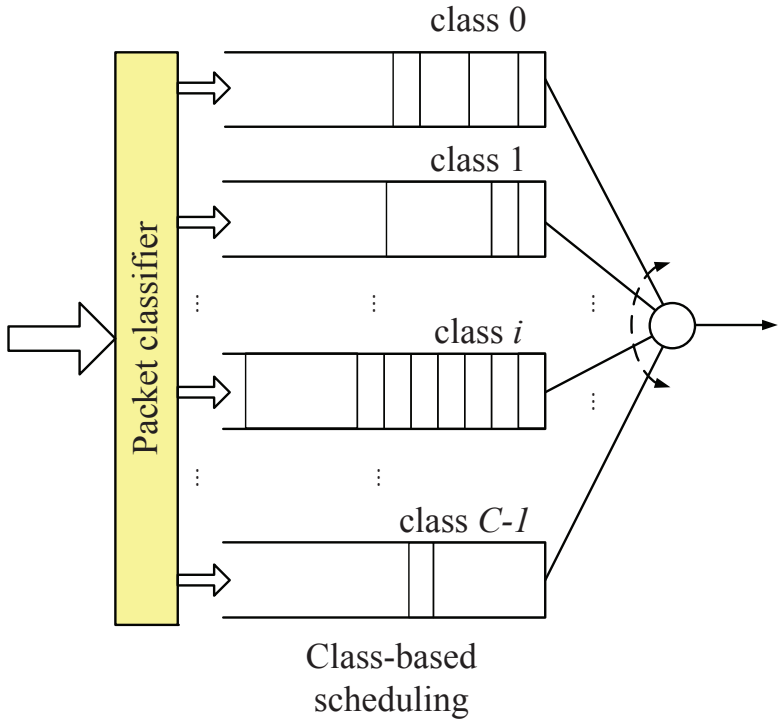


Figure 3.2: Demonstration of a class-based scheduling system. Variable-length packets are stored into queues based on their QoS classes.

classes have different requirement on delay or bandwidth. As mentioned earlier, different services have different QoS requirements, i.e. some traffic is sensitive to delay while others require adequate bandwidth. Based on this model, a separate queue for each QoS class is created in a switch to store packets of the same class. By giving different priorities, different shares of bandwidth are assigned to the classes.

However, this traditional queuing scheme is incapable of providing isolation to traffic flows, which have the same QoS classes but different sources or destinations. Since all the packets of the same QoS class are stored in the same queue, a malicious flow can consume a huge amount of bandwidth, resulting in a significant performance degradation to other flows of the same class. Given the requirement of flow isolation, the

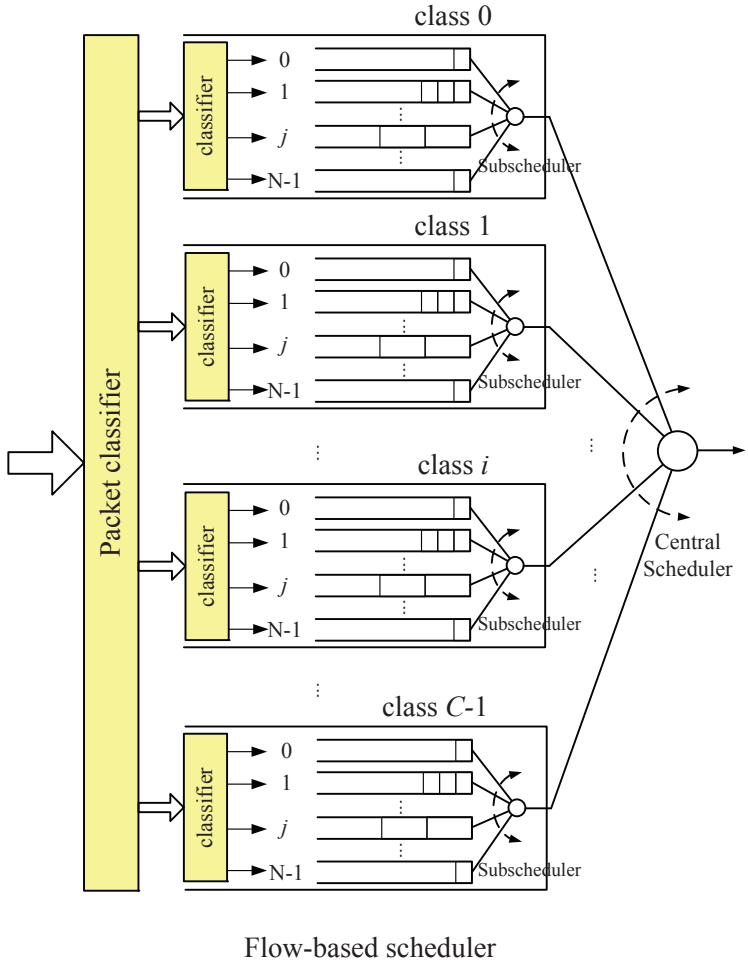


Figure 3.3: Demonstration of a flow-based scheduling system. Variable-length packets are first sorted based on the QoS classes, then further sorted based on the flow ID.

flow-based scheduling scheme is proposed in [1], as shown in Figure 3.3.

Within each class queue, a separate queue is assigned to packets of the same source or destination, depending on whether the flow-based scheduler is located at the output port of input port. In addition to

using $\langle Source|Destination|QoS\ class\rangle$ to define a flow, information such as VLAN ID, can also be included. A central scheduler grants permission to each subscheduler using DRR, and each subscheduler selects packets from different queues in a DRR manner, once a permission granted. This architecture ensures isolation between each flow as well as each class. However, to provide end-to-end QoS guarantee, all the nodes in the network should be upgraded to this advanced architecture, which is not cost-efficient.

3.3 System Model and Problem Definition

As discussed in Section 3.2, although the flow-based scheme can provide the operator with a network with end-to-end QoS guarantee by replacing all the switches with advanced models, at the same time it places a considerable burden on the network operator, especially when the size of the network is fairly large. Such a process of upgrading a large network inevitably requires a long deployment time and a substantial financial investment. Distributing intelligence, in terms of large size of memory, advanced scheduling algorithm, flow control ability and so forth, to all the nodes in the network will inevitably need a management platform that can manage and configure the switches efficiently. Besides, the resources, e.g. the size of the buffer, which the operator brings to each of the nodes, may not be fully utilized.

A possible alternative is to centralize intelligence and introduce an intelligent switch with the knowledge of the network topology located at the edge of the network. One good example could be the ingress and egress router in an MPLS network. Typically, the MPLS label is attached to an IP packet at the ingress router and removed at the egress router, while label swapping is performed at the intermediate routers. This intelligent node should be able to manage the traffic on behalf of other nodes which lack advanced traffic management ability, and thus can avoid potential traffic congestion in the network.

From an IPTV traffic distribution's point of view, the tree topology is usually used to construct the network [54,55]. A generic tree topology in Figure 3.4 is presented.

The intelligent node is connected to the root node, denoted as *HS*. The tree network is assumed to have N levels and each node is a parent

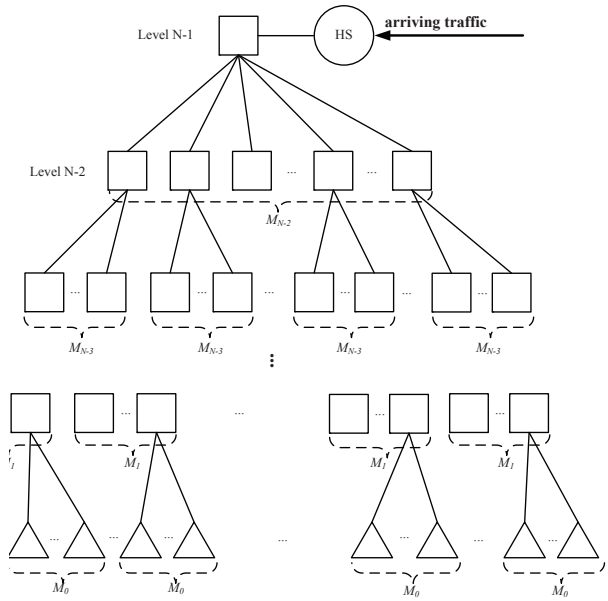


Figure 3.4: A balanced tree topology network with the topology-based hierarchical scheduler (HS) node connected to the root node.

to the nodes in the level below and at the same time a child to the node in the level above. Nodes in level *0* are leaf nodes and have no child nodes attached. It is assumed that in each level the number of child nodes connected to an upper-level node is the same, denoted as M_l, where $l = 0, 1, 2, \ldots N-1$. Except for the root node, *HS*, a random node in the network is denoted as N_i^l, where $0 \leq i < \prod_{j=0}^{N-1} M_j$, and $M_0 = 1$. The transmission speed of the link between a node N_i^l and one of its child nodes is assumed to be $\frac{1}{M_l}$ of the link speed between N_i^l and its parent node. Since the link speed of a node is evenly divided and allocated to its child nodes, this network topology is referred to as the *balanced tree* topology in this chapter.

Since the nodes in the network lack advanced traffic management functionalities, the edge node *HS* should schedule the arriving traffic on behalf of the lower-level nodes. The topology-based hierarchical scheduling scheme is an attractive candidate for such a situation.

3.4 Topology-based Hierarchical Scheduling Algorithm

First, the principle of DRR is reviewed before describing the topology-based hierarchical scheduling algorithm. To serve queues, the scheduler uses round-robin pattern with a *quantum* assigned to each queue, which is the number of bytes allowed to be sent from a queue within one round. The quantum size in bytes, Q, is usually set to the maximum packet size to ensure that at least one packet is served during one scheduling round to maintain a low complexity. If Q is larger than the length of the packet in bytes, L, in the queue, the packet is sent and $Q = Q - L$. If a queue is not able to send a packet in the previous round due to too large packet size, i.e. $Q < L$, the remainder from the previous quantum will be added to the quantum for the next round. Hence, the deficits are kept and unfairly treated queues are compensated in the next round. By adjusting the quantum size for each queue, the total bandwidth is allocated in proportion to the quantum size.

To provide per-flow isolation, the flow-based schedulers are leveraged to compose the topology-based hierarchical scheduling system. Based on the balanced tree topology of the connected network, a mapping can

be created by several token schedulers. Figure 3.5 demonstrates the schematic structure of the topology-based hierarchical scheduler.

A token is generated for each arriving packet at the packet classifier. The token should carry scheduling information for its corresponding packet, such as packet weight, destination/source ID, and QoS class. The packet weight is a value in proportion to the actual packet length. It is used by the token scheduler as a virtual packet length to control the packet transmission rate. The packet is forwarded into the packet memory and the token is stored in the token queues based on its flow ID $\langle Source|Destination|QoS\ class\rangle$.

The topology-based hierarchical scheduling algorithm contains 3 steps, *Selection*, *Grant*, and *Update*, which are described as below:

Selection: Using the DRR scheduling algorithm, the scheduling system establishes N levels of token schedulers, from level 0 to $N-1$. Each scheduler, except the top-level scheduler S_{N-1}, and each token queue has a Deficit Counter (DC) attached to store the value of Q, the remainder of the quantum size. For a token queue, if it is not empty during a scheduling period, it is defined to be *backlogged* [33]. Similarly for a scheduler, it turns backlogged only when it has selected a queue or a lower-level scheduler to serve. A level-p scheduler $S_p(x)$, $0 < p < N-1$, operates the DRR algorithm on its backlogged level-(p-*1*) schedulers. A level-*0* scheduler, $S_0(x)$, runs the DRR algorithm on its backlogged class schedulers cS_j, and each class scheduler executes the DRR algorithm on the backlogged token queues. Following this process, the schedulers make the scheduling decision level by level until level-(N-*2*) schedulers complete the selection phase. All the decisions made in this step are pending and wait for further grants from the top-level scheduler.

Grant: A scheduler can grant a permission to the selected token queue/scheduler *when and only when* it receives a grant from its upper-level scheduler(parent scheduler). For the top-level scheduler, S_{N-1}, it has no parent scheduler and therefore it only grants permissions using the DRR algorithm. Once a permission is granted by S_{N-1}, it is passed level by level until the permission reaches a token queue. A token path is established after this step.

Update: Upon the reception of the permission, a token is sent to S_{N-1} along the token path and all the scheduler on the path update their deficit counters with a reduction of the packet length information

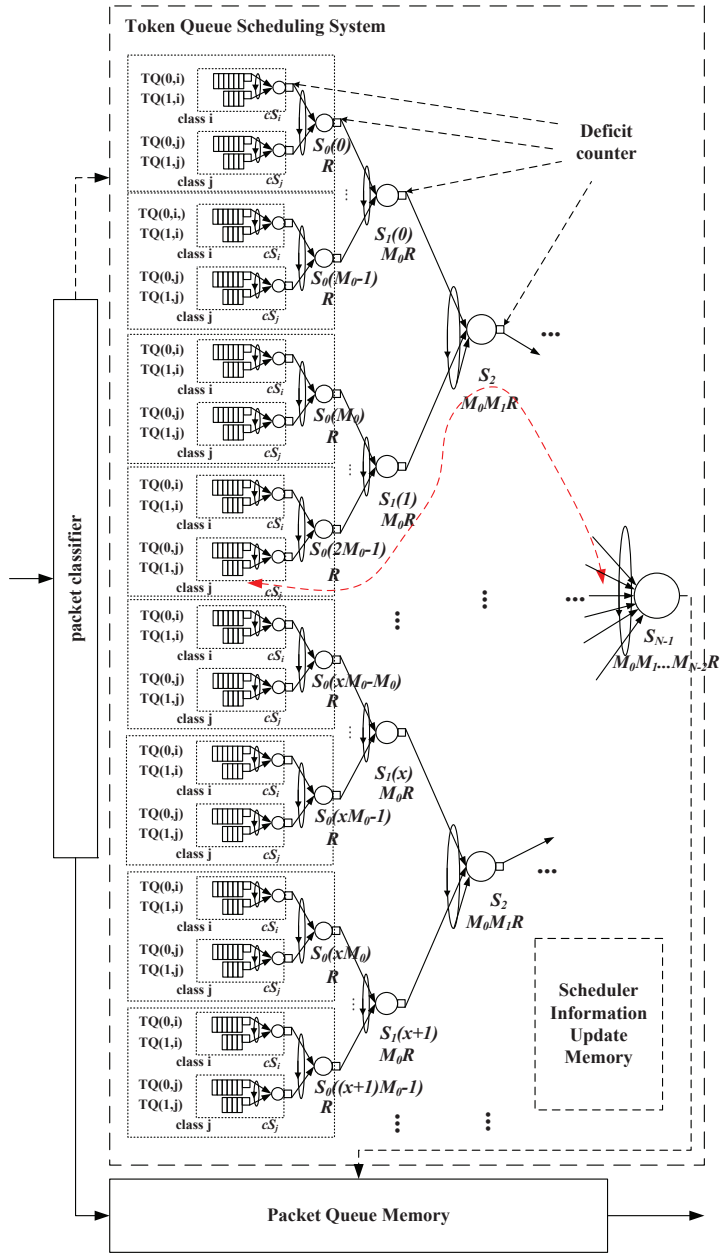


Figure 3.5: The topology-based hierarchical scheduling system. The main scheduler S_{N-1} grants permission backwards, and a token path is established, shown as the red arrowed dash line. Token(s) is passed to S_{N-1}, and deficit counters on the path are updated.

carried by the token, i.e. $DC = DC - L$. When S_{N-1} receives the token, it sends out the correspondent packet from the packet memory according to the information carried by the token. After this step, a new scheduling round begins until the quantum sizes of all token schedulers become zero or no selection is made because the packet size is larger than the quantum size.

To avoid congestions from occurring in the network, token schedulers control the packet transmission rate by the token rate and the packet weight. Packet Weight (PW) is a function of the actual packet length l, and is calculated by the packet classifier for each arriving packet. In the token scheduling system, a virtual packet transmission time is calculated as the PW divided by the token rate. The description PW function is shown in Equation 3.1.

$$PW(l) = l \cdot f_w. \tag{3.1}$$

where l denotes the packet length stored in the Ethernet Media Access Control (MAC) header, and f_w is the configurable packet weight factor.

Since the token rate corresponds to the actual transmission rate of the node in the network, it is calculated as a product of the weight factor f_w and the actual link rate R_{link}. The description of the token rate calculation is shown in Equation 3.2. The network operator can modify the packet weight factor to control the granularity of token rates in order to adapt the switch to the network.

$$R = f_w \cdot R_{link}. \tag{3.2}$$

where R is the basic token rate and R_{link} is the link rate of the end node.

The time between two token selections is the sum of the packet weight of the token(s) divided by the token rate of the scheduler. By this mean, the packet transmission rate is controlled so that packets are transmitted within the capacity of the nodes in the network, and thus traffic congestion can be avoided.

Each S_0 can only send one token to S_{N-1} per time unit. S_{N-1} is able to grant up to $\prod_{i=0}^{N-2} M_i$ permissions to guarantee at most each S_0 receives a permission within one time unit. A token scheduler $S_p(x)$ on level p at most receives $\prod_{i=0}^{p-1} M_i$ permissions from its parent scheduler

on level $p+1$. Therefore, if the token rate of S_0 is assumed to be R, the token rate of S_p becomes $\prod_{i=0}^{p-1} M_i \cdot R$, $0 < p \leq N-1$.

Schedulers on different levels have different quantum size, which corresponds to their token rate. If assume that the quantum size for a token queue is Q, then the quantum sizes for the class scheduler and S_0 are both Q because a level-*0* scheduler can only send one token per time unit. For a level-p scheduler S_p, the quantum size becomes $Q_p = \prod_{i=0}^{p-1} M_i \cdot Q$, $0 < p < N-1$.

It is important to mention that the structure of the topology-based hierarchical scheduler is not limited to the example shown in Figure 3.4, but can be reconfigured according to the actual network topology. If the topology is an asymmetric tree or a star for instance, the token schedulers will be reorganized and the logical structure will be configured accordingly. For the unbalanced or non-binary tree topology, where bandwidth is not evenly divided, the scheduling system can adjust the quantum size of the token schedulers on each level accordingly to match the bandwidth allocation. The quantum size of a token scheduler on level-p is actually the sum of the quantum sizes of all its child token schedulers, $Q_p(x) = \sum_i Q_{p-1}(i \rightarrow x)$, where $i \rightarrow x$ implies that $S_{p-1}(i)$ is one child token scheduler of $S_p(x)$.

The hierarchical scheduler is a linear combination of several DRR schedulers. Each DRR scheduler has a time complexity of $O(1)$ [33]. If the topology-based hierarchical scheduler has N hierarchies or levels, it needs to establish a token path in N steps and thus the time complexity will be $O(N)$. When $N = 1$, the scheduler will become a single DRR scheduler of which the time complexity is $O(1)$.

3.5 Simulated Performance

The comparison of the QoS performances is carried out between the topology-based hierarchical scheduling, the flow-based scheduling, and the class-based scheduling algorithms, in terms of delay, jitter, and flow protection in this section. The simulations are carried out in OPNET Modeler [56].

The network configuration for the simulation is shown in Figure 3.6. Three networks of the same balanced tree topology are created, each of which uses one scheduling scheme, i.e. the class-based, the flow-

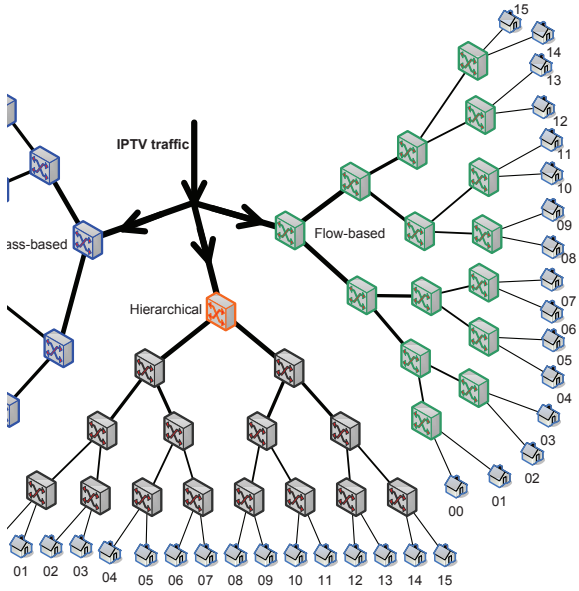


Figure 3.6: The simulation scenario set-up. Three networks, with different scheduling schemes, of the same binary tree topology are connceted to the same traffic source.

based, and the topology-based hierarchical scheduling. Each network is assumed to have 5 levels in total, including the top-level node and the end nodes. It is assumed that, for each network, $M_l = 2 \quad (l = 1, 2, 3, 4)$. The number of leaf nodes thus become $2^4 = 16$. All three networks are connected to the same IPTV traffic generator, which provides 16 identical traffic flows simultaneously.

3.5.1 Evaluation of Statistical Multiplexing Gain

Each flow is configured to be transmitted to one different end node. The peak and the minimum bandwidth of each flow is assumed to be 10 Mbps and 4 Mbps, respectively. The percentage of the peak bandwidth is assumed to be 50%, resulting in an average bandwidth of $\bar{B} = 7$ Mbps. The output link rate of the edge node is reduced while the input traffic flows are maintained in order to evaluate the Statistical Multiplexing Gain (SMG) performance [57, 58]. The input-output rate ratio is used

as the x-axis. Since the output rate is reduced gradually, the ratio increases from the initial value of 1.0.

Due to the burstiness of the input traffic and the aggregation of flows, the capacity of the link can be saved by using statistical multiplexing to reduce the link rate. If the traffic is not highly bursty, the average end-to-end delay and jitter will increase as the link rate decreases. Figure 3.7 provides the average end-to-end delay comparison between the class-based, flow-based and hierarchical scheduling under various input-output rate ratios. Figure 3.8 shows the jitter comparison under the same range of input-output rate ratio.

In Figure 3.7, hierarchical scheduling has improved the performance on the average end-to-end delay. The curve of hierarchical scheduling is below the other two, class-based and flow-based scheduling schemes. To achieve the same end-to-end delay, hierarchical scheduling can sustain a higher reduction of link capacities. As the input-output rate ratio increases, the average end-to-end delay increases for all three schemes but the slope of the hierarchical scheduling curve becomes lower than the class-based scheme.

In Figure 3.8, the three scheduling methods, i.e. class-based, flow-based and hierarchical scheduling have shown alike performance, in terms of traffic jitter under different input-output rate ratios. As the input-output rate ratio increases, the jitter of all three schemes become larger. Hierarchical scheduling has little improvement on the jitter performance.

It can be obtained from the results that the improvement of the SMG factor by the hierarchical scheduling scheme is limited. The traditional way of browsing websites allows the operator to reduce the required bandwidth for the aggregated flows. If there are 1000 users, for instance, and each one is guaranteed 10 Mbps download bandwidth, the operator can assign 200 Mbps bandwidth for the aggregated traffic to satisfy the requirement, since not all the users need the resource at the same time. The SMG thus becomes $\frac{1000 \cdot 10}{200} = 50$ under this circumstance. When IPTV services are introduced in a network, the SMG factor will begin to decrease, due to the fact that, the traffic is low in burstiness but high in bandwidth consumption. The traffic characteristic is different from those applications that generate bursty traffic, such as website browsing.

The advantage of the hierarchical scheduling scheme is that it can

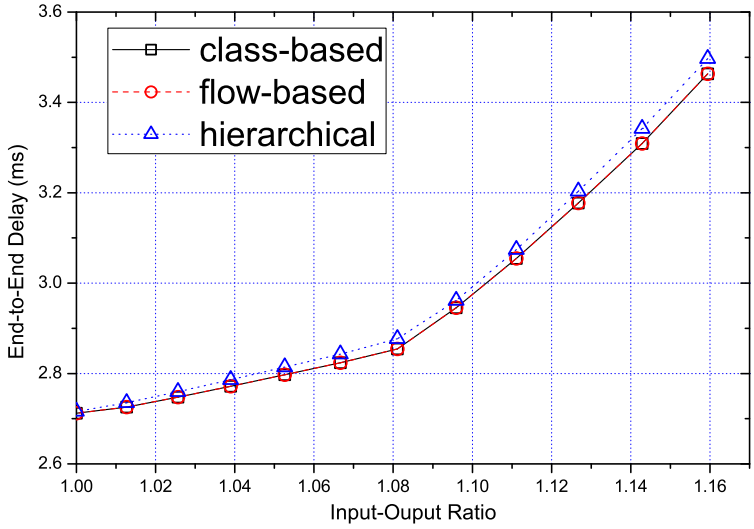


Figure 3.7: Comparison between class-based, flow-based, and hierarchical scheduling schemes in terms of average end-to-end traffic delay under different input-output rate ratio

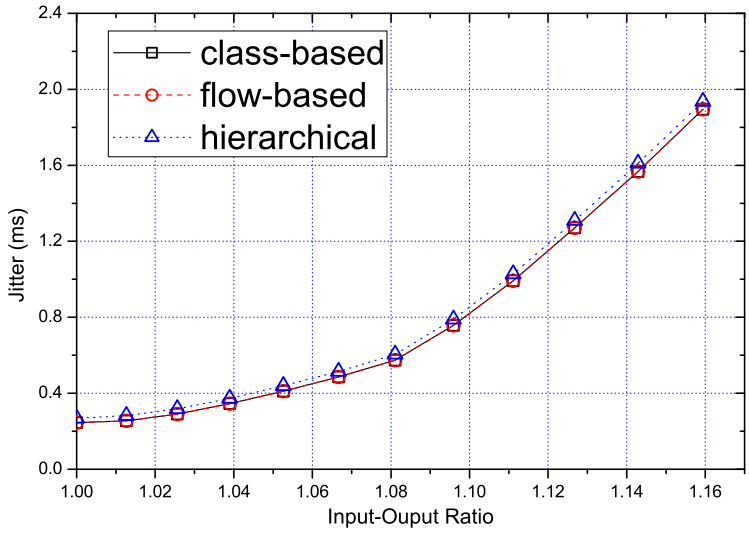


Figure 3.8: Comparison between class-based, flow-based, and hierarchical scheduling schemes in terms of traffic jitter under different input-output rate ratio.

provide nearly the same performance as the distributed intelligence fashion. By learning the network topology through the management plane or manual configuration, the scheduler at the edge of the network forms a mapping structure with virtual token schedulers. The cooperation between each token scheduler is far more efficient than the cooperation between different nodes. The centralized intelligence way of traffic management can be considered as a solution.

3.5.2 Evaluation of Flow Protection

A conforming traffic flow is configured to have an average bandwidth of 9 Mbps, which is similar to the bandwidth needed by a high definition IPTV channel. 16 flows respectively bound for 16 destinations are sent to the three networks simultaneously. The link speed is reduced by half for each level as explained previously. For the end user, the link supports up to 10 Mbps transmission rate.

To evaluate the flow protection and isolation ability of the networks, a highly bursty traffic flow is introduced for a certain period of time.

The impact to the conforming flow is then observed at the destination. The highly bursty traffic flow has a higher average bandwidth than a conforming flow.

The simulation lasts for 60 seconds and the highly bursty traffic flow is introduced from 10 to 20 seconds. In the networks shown in Figure 3.6, the highly bursty flow is bound to user 01. The flow to user 00 is observed because it is the most affected by the highly bursty flow.

In Figure 3.9 the comparison between class-based, flow-based and hierarchical scheduling under the malicious flow attack is presented. The bandwidth of the highly bursty traffic is 9.5% more than the normal flow. Since the class-based scheduling scheme cannot distinguish different flows of the same traffic type, the normal flow is affected the most in terms of increase in end-to-end delay. Flow-based and hierarchical scheduling schemes are both capable of flow isolation, and thus the end-to-end delay of the normal flow increases slightly. Class-based scheduling scheme, under the malicious flow attack, performs the worst, and thus the comparison will be carried out between the flow-based and the hierarchical scheduling schemes.

In Figure 3.10, the bandwidth of the highly bursty traffic flow is increased to be 67% more than a normal flow bandwidth. The affected end-to-end delay of the conforming flow bound to destination 00 is presented. A comparison between the flow-based scheduling and the hierarchical scheduling is presented in this figure. The highest end-to-end delay of the flow-based scheduling network is increased up to around 4.5 ms, while the delay of the network using hierarchical scheduling scheme is increased to around 3.0 ms at most. After the highly bursty flow stops, both end-to-end delays are restored to the normal level. The hierarchical scheduling obviously has better performance than the flow-based one in terms of flow protection.

To further investigate the flow protection and isolation ability of the two scheduling schemes, i.e. flow-based scheduling and hierarchical scheduling, several simulations under different traffic load of the highly bursty flow are carried out. The average end-to-end delay of the affected period, during which the highly bursty flow is introduced, is measured for each circumstance. The comparison results are shown in Figure 3.11. The average bandwidth of the highly bursty flow bound to destination 01 is increased from 10 to 16 Mbps. Both two schemes show similar

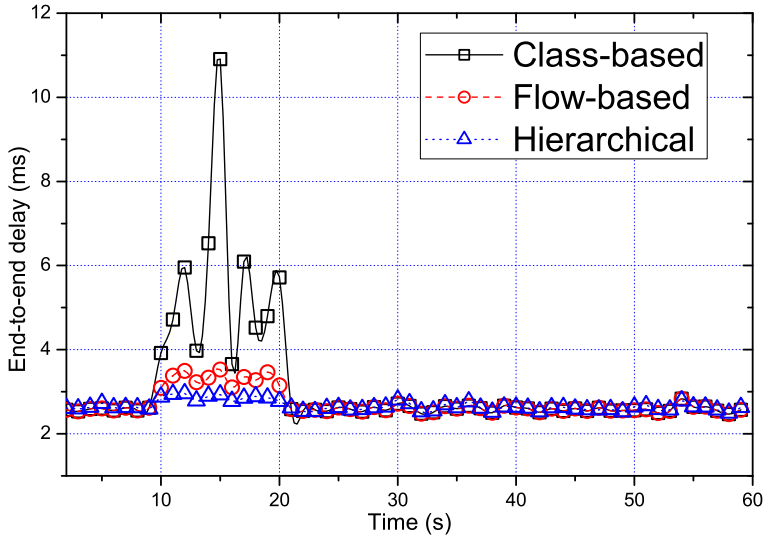


Figure 3.9: Comparison between class-based, flow-based and hierarchical scheduling in terms of traffic delay when a non-conforming flow appears. Bandwidth of the highly bursty traffic is 9.5% more than a normal flow.

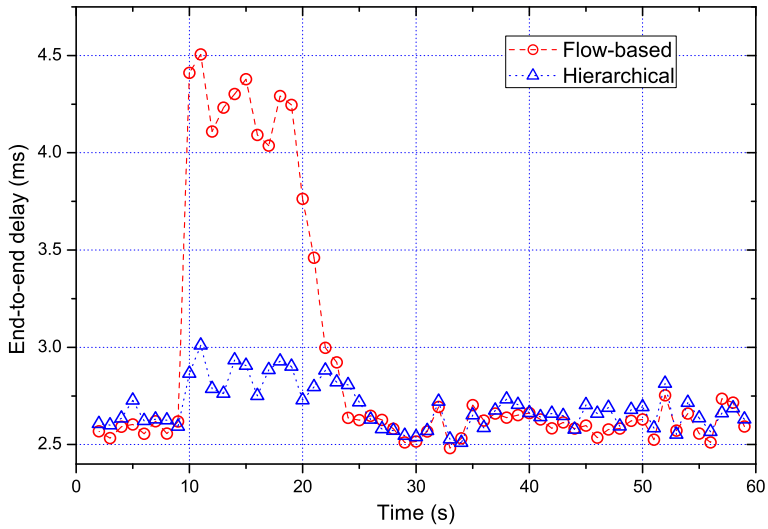


Figure 3.10: Comparison between flow-based and hierarchical scheduling in terms of traffic end-to-end delay when the load of a highly bursty flow increases. Bandwidth of the highly bursty traffic is 67% more than a normal flow.

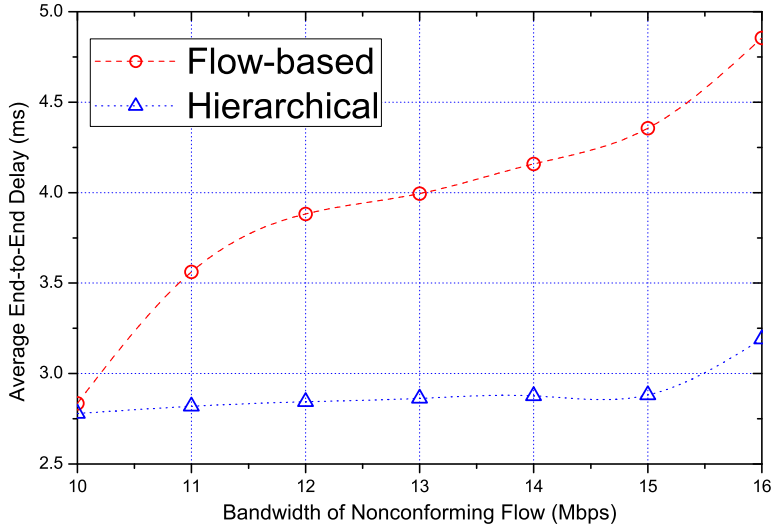


Figure 3.11: Comparison between flow-based and hierarchical scheduling in terms of average traffic delay of the affected period as the load of a highly bursty flow increases.

average end-to-end delay under the bandwidth of 10 Mbps. This is because the switches in both networks still have enough capacity. Once the highly bursty flow increases the bandwidth more than the maximum limit, congestion will occur and consequently cause an addition to the average end-to-end delay of the normal flow. The curve of the hierarchical scheduling scheme, compared to the flow-based one, remains stable, which indicates that the hierarchical scheduling scheme is able to provide better flow isolation and protection.

The improvement should be credited to centralizing network intelligence in the edge node. Potential congestion or any malicious attack is handled by the scheduler inside the node. Necessary internal resources are arranged and utilized by the node to diminish the bad behavior. On the other hand, the flow-based scheme, which is a distributed way to protect flows, could be ineffective or inefficient since the cooperation between each node in a network is more difficult than the cooperation between each scheduler in the hierarchical scheduler. From the point of view of protecting traffic flows to guarantee the requirement of QoS,

the hierarchical scheduling scheme shows better performance than the distributed flow-based scheme.

It is also worth mentioning that the results have shown a trend of how a flow is affected by highly bursty traffic in a network using various scheduling schemes. In a real network, the actual values will be very likely to differ from the ones shown in these figures. What is important is the relative relation demonstrated by the results.

3.6 Summary

In this chapter, a topology-based hierarchical scheduling scheme for IPTV traffic management in Carrier Ethernet transport networks is proposed. The hierarchical scheduler can be placed at the edge of broadband access network, where the topology is relatively static from an IPTV distibution's point of view.

Based on the assumption that the topology-based hierarchical scheduler is able to acquire the network topology, it has demonstrated a method where the hierarchical scheduler combines several DRR token schedulers to build a mapping structure of the connected network. The hierarchical scheduler manages traffic on behalf of other nodes in the network and is able to avoid severe performance degradation from the attack of maliciously behaving traffic flows.

Simulation results have shown that the proposed scheduler can provide a better flow protection and isolation against potential attack from malicious traffic and as a result provide QoS guarantee, which is a significant requirement for IPTV services in Carrier Ethernet transport networks. The proposed scheme could also bring benefit to network operators in terms of deployment effort and cost-efficiency.

It is also important to mention that the hierarchical scheduling scheme presented in this chapter is not limited to the topology used in the example. As a matter of fact, the scheduler can adapt to different network topologies. By network management or manual configuration, the scheduler can know where the potential congestion points are and how the network topology is. Different knowledge about the network leads to different combinations of the DRR token schedulers. The flexibility thus enables the scheduler to adapt to various network topologies, e.g. star, asymmetric tree and so forth. It is out of the scope of this

chapter to discuss how the combination of the DRR token schedulers is implemented.

Chapter 4

Multicast Scheduling Algorithms for Input-Queued Switches

The Input Queuing (IQ) architecture has been favored for designing multicast high-speed switches due to its scalability and low implementation complexity. Various existing improvements on the First-In-First-Out (FIFO)-based IQ architecture have been proposed to reduce the Head-Of-Line (HOL) blocking problem and as a result to increase throughput. However, a trade-off exists between the complexity and the performance of the multicast scheduling algorithms. Algorithms with low implementation complexity usually suffer from the HOL blocking [59]. On the other hand, algorithms that achieve high throughput usually are high in implementation complexity, making them hard to scale in terms of either switch size or port speed [60,61]. Given that multicast switches are able to reduce the continuously increasing network load, an effective and efficient, yet low-complexity multicast scheduling algorithm is in need.

In this chapter, the Multi-Level Round-Robin Multicast Scheduling (MLRRMS) algorithm is presented for FIFO-based input-queued switches. First of all, the advantages of the IQ architecture are discussed in comparison with other architectures. Different algorithms developed for the IQ architecture are shortly reviewed as the background knowledge for the MLRRMS. The problems encountered by the system architecture are defined, and the solutions to the problems are provided, which

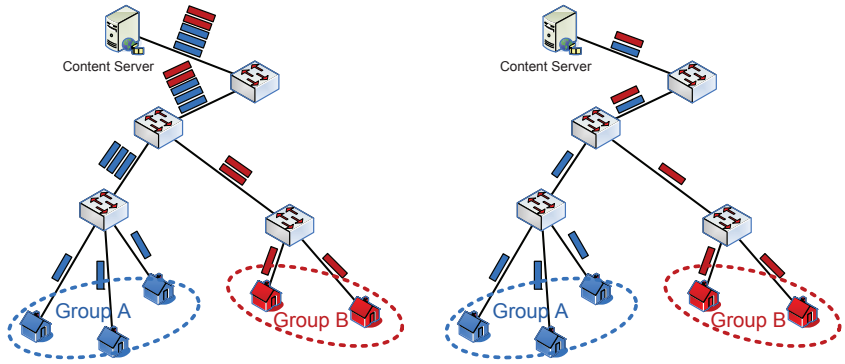
comprise the MLRRMS algorithms. Analytical analysis and simulated performance results demonstrate that the FIFO-based IQ multicast architecture is able to achieve significant improvement with the MLRRMS algorithm in terms of multicast delay and throughput with the capability of searching a limited number of cells stored into the input queues.

4.1 Introduction

It is foreseeable that network capacity will increase substantially as bandwidth-intensive applications become more and more popular and the required network bandwidth will grow correspondingly. Traffic generated by the bandwidth-intensive applications, such as videoconferencing and IPTV, usually have several groups of subscribers and each group subscribes to the same content, e.g. group A watches a football game and group B watches a movie.

Even though it is realizable to complete a transmission of a content to a group by several unicast flows, i.e. to transmit M traffic flows to the network with each bound for a subscriber in the target group that has M subscribers, the traffic load in the network will incontestably skyrocket. Multicast, on the other hand, is able to reduce the traffic sent to the network. As demonstrated in Figure 4.1, instead of loading the network with redundant unicast traffic as in Figure 4.1(a), the switches in Figure 4.1(b) are able to copy the received packet and send them to the subscribed nodes and thus the traffic load in the network is substantially reduced. The spared network resources can be used for other services. As a result, multicast-enabled nodes are favored for those high-bandwidth services in order to reduce the required bandwidth and the multicast latency in the transport network, e.g. the Carrier Ethernet transport network.

Due to the fact that fixed-size switching technology is able to achieve high switching efficiency, it is considered widely in literature [7, 29, 60–68]. Variable-length packets are segmented into fixed-size *cells* before traversing the switch fabric, and are reassembled back into packets before being sent out of the switch. As a matter of fact, in several advanced Internet routers/switches and prototypes, the switch fabric internally operates on cells, such as the Cisco GSR [69], the Tiny-Tera [70], and the iPoint [71]. In the rest of this section, *packet* is used as a generic



(a) Illustrating of Unicast. (b) Illustration of Multicast.

Figure 4.1: A simple explanation of using unicast and multicast to provide IPTV services.

term to indicate data unit regardless of the length, for simplicity.

To ensure a low packet loss rate, switches usually have buffers installed to store packets that cannot be served immediately on their arrivals. Buffers can be placed at the input port, at the output port, or in a location shared by input and output ports. Based on the position of the buffers, buffering mechanisms of switches can be mainly categorized into several types: Input Queuing (IQ), Output Queuing (OQ), and shared-buffer. Different combinations of these schemes are possible in the practical switch design.

Pure IQ switches place FIFO queues at the inputs as illustrated in Figure 4.2. The memory only runs as fast as the input line speed, which lowers the implementation complexity, but the IQ scheme with FIFO queues suffers from degraded throughput due to the HOL blocking problem [59], where a packet failing to compete for the output ports will stay at the head of queue and blocks those behind to be transmitted, even if their destined output ports are available. The advantage of the IQ scheme is that it can easily scale up in terms of switch size and link speed, but HOL blocking limits the throughput to approximately 58.6% of the maximum [59].

Pure OQ switches set buffers at the outputs to store packets, as shown in Figure 4.3. As a result, packets received by an OQ switch can

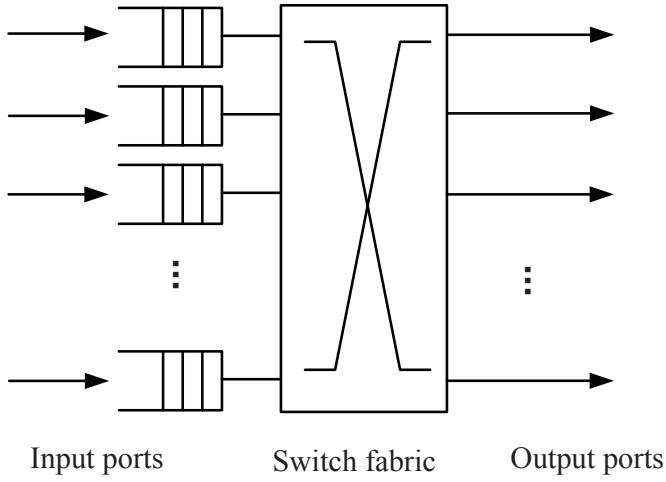


Figure 4.2: Illustration of an input-queued switch. A pure input-queued switch places FIFO buffers at the input ports. The buffers need to run only as fast as the input links, but the head-of-line blocking can be endured and cause throughput degradation.

always reach their destination ports immediately on their arrivals, given the condition that the buffer runs N times the link speed for a switch with N input ports in the worst case that packets at all the input ports are destined to the same output port, which eliminates the HOL blocking problem. However, the scalability of the OQ architecture is constrained. Since no input buffers are allocated, the switch must deliver N packets to an output buffer to avoid packet loss, and that output buffer must be able to store N packets in the time it takes for one packet to arrive at an input. This buffer speedup requirement limits the scalability of the OQ scheme.

In the shared-buffer architecture, input ports and output ports share a memory pool as shown in Figure 4.4. Incoming packets are stored in the shared memory. The packet headers are extracted and used for scheduling purpose by the switch. When a packet is scheduled for transmission, the output port removes it from the shared memory. However, for an $N \times N$ switch, the switch must be able to read and write N packets in only one packet arrival time, which strongly restricts the scalability

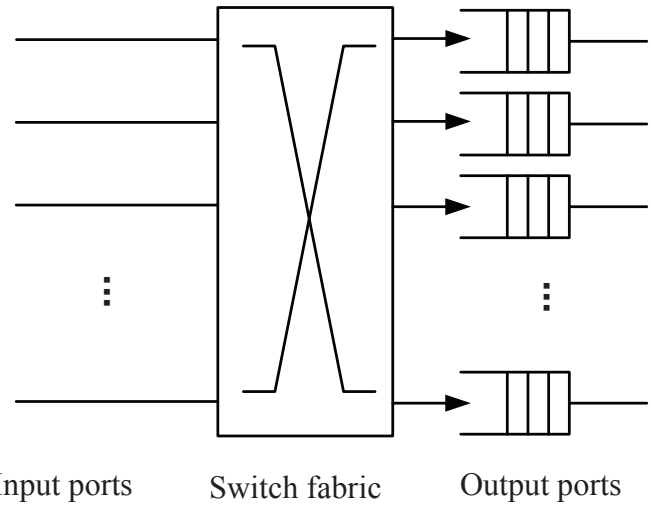


Figure 4.3: Illustration of an output-queued switch. A pure output-queued switch allocates buffers only at the output ports. The buffers and the switch fabric need to run N times as fast as the link speed in order to avoid packet loss. This speedup requirement limits the scalability of the output-queuing scheme.

of the switch.

Since the advantage of the IQ architecture surpasses the others, in terms of building a scalable architecture for high-speed switching, the IQ scheme is favored except for its HOL blocking problem when FIFO queues are employed. By using a different buffering strategy at each input port, the HOL blocking can be eliminated entirely. This is known as Virtual Output Queuing (VOQ), where each input maintains a separate queue for each output [72, 73], as shown in Figure 4.5. Since a packet cannot be blocked by a packet ahead of it which is bound for a different output port, the HOL blocking is thus eliminated. No speedup is required in the VOQ scheme because, for cell switching, at most one cell can arrive and depart from each input within a cell transmission time slot. Several scheduling algorithms are proposed based on the VOQ architecture, such as iSLIP [74] and PIM [72], providing a solution to eliminate the HOL blocking problem and achieving higher throughput than the IQ architecture with FIFO queues at the inputs [73]. Strictly speaking, the VOQ is a subcategory of the IQ architecture, where buffers

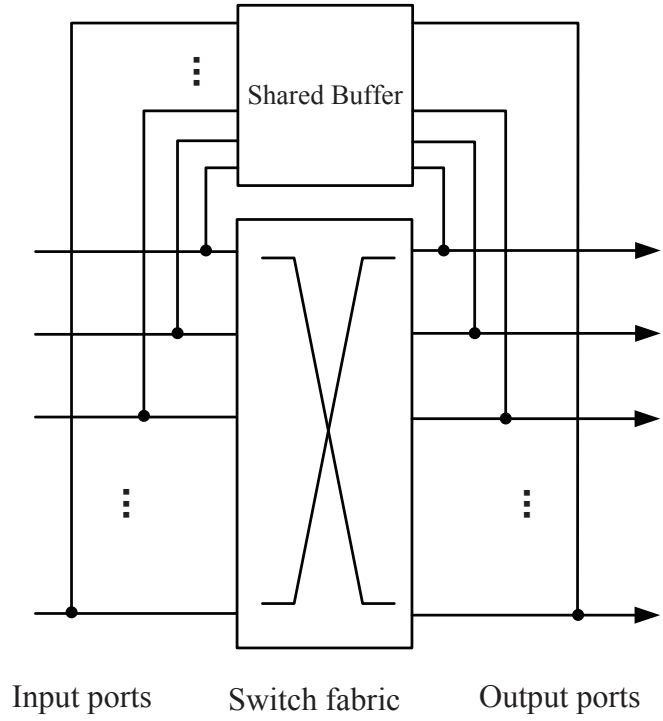


Figure 4.4: Illustration of a shared-buffer switch. Input and output ports share a memory pool, where arriving packets are stored. The memory must be able to read and write N packets in one packet arrival time for an $N \times N$ switch. This speedup requirement restricts the scalability of the shared-buffer scheme.

are allocated at the input ports. But for simplicity, we use the term *IQ* to refer to the IQ architecture with FIFO queues and *VOQ* for the IQ scheme where separate queues are employed at each input for unicast.

Although VOQ, by creating separate queues in each input, can entirely eliminate the HOL blocking and improve the throughput performance, it scales poorly due to the requirement for N^2 queues in total for an $N \times N$ switch. The scalability of the VOQ mechanism becomes even worse when applied to multicast. To eliminate the HOL blocking for an $N \times N$ multicast switches, each input must maintain a separate queue to store the multicast packets of each possible combination of N destinations. Such architecture is called MultiCast Virtual Output Queu-

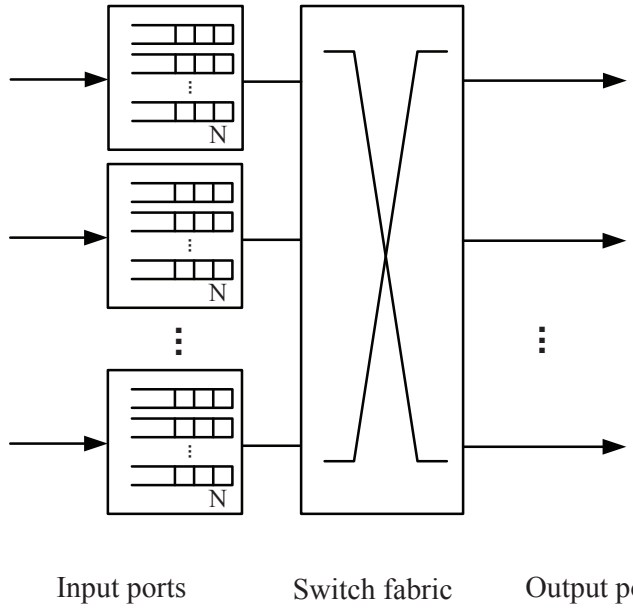


Figure 4.5: Illustration of an virtual output queued switch. The head-of-line blocking can be eliminated entirely by employing a separate queue for each output at each input.

ing (MC-VOQ) [75]. MC-VOQ requires $2^N - 1$ queues for each input and thus in total $N \cdot \left(2^N - 1\right)$. The scalability of the MC-VOQ architecture is poor and thus becomes impractical in medium/large switches. For simplicity, we use the term MC-VOQ to refer to the VOQ architecture for multicast throughout this chapter.

Therefore for multicast switches, the attention is focused on the scalability of the IQ architecture, where the arriving multicast packets are stored in a FIFO queue at each input. No buffers are allocated to the outputs or shared between the input and output ports in order to avoid the requirement of speedup. To reduce the HOL blocking problem of using FIFO queues for multicast traffic, a novel multicast scheduling algorithm, MLRRMS, is proposed. The MLRRMS is implemented in a distributed manner to provide high scalability instead of using a centralized scheduling module, which can hinder the scalability of high-speed switches.

The rest of this chapter is structured as follows. Section 4.2 briefly introduces the related works in the multicast scheduling algorithms for completeness as background knowledge. Section 4.3 describes the system architecture used throughout this chapter, and defines problems to be solved. In Section 4.4, the MLRRMS algorithm is proposed and described in detail. In Section 4.5, the analysis of the MLRRMS algorithm is provided. In Section 4.6, simulations and discussions on the results are presented. Finally, Section 4.7 concludes this chapter.

4.2 Related Work

Since it is impractical to use the MC-VOQ architecture for multicast where each destination combination requires a queue, several architectures and algorithms have been proposed to schedule multicast traffic leveraging either the FIFO or the VOQ architecture.

The multicast scheduling algorithm for IQ switches, also know as TATRA [62], focuses on the IQ architecture for multicast, where multicast cells are stored in FIFO queues. After the scheduler decides which cells to send, it leaves a residue of cells to be scheduled in the next cell time. Motivated by the game Tetris, TATRA schedules the residue of cells based on the departure date, which is the number of cell times before a copy of the cell is served. The TATRA algorithm is strict in fairness and achieves low latency, however, it is high in implementation complexity. To remedy this, another algorithm, the Weight-Based Algorithm (WBA), is proposed in [62] as a replacement to TATRA due to its simplicity. This algorithm works by allocating weights to input cells according to their age and fan-out (number of destinations in the multicast group) at the beginning of every cell time, and each output port choosing the HOL cells with the highest weights. Although the WBA ensures fairness and has a low implementation complexity, it suffers from the HOL blocking problem.

The FIFO-based Multicast Scheduling (FIFOMS) algorithm [60], and the Credit based Multicast Fair (CMF) scheduling algorithm [61] utilize the VOQ architecture for unicast to schedule multicast traffic. Instead of assigning a queue to each combination of N destinations, only N queues are allocated for each input port. Up to N address tokens are generated for each arriving cell, each of which is stored in the

a queue corresponding to a destination. The arrived multicast cell is stored in a memory pool and is linked by its address tokens. Based on the scheduling decisions from the scheduling algorithms executed on the address tokens, the multicast cell is sent and is removed from the memory until all its destinations are reached. The FIFOMS and CMF are able to achieve low latency and high throughput, but the bottlenecks of the architecture can hinder its scalability.

The hardware complexity of the address token generator can be $O(N)$, since up to N tokens are generated for each arriving cell, and the address token generating rate is required to be N times the cell arrival rate due to that multiple tokens are generated for each arriving cell within one cell transmission time. Besides, this architecture requires a complex buffer management mechanism to send a multicast cell using the link address in an address token because the actual cell to be sent is not always the HOL cell. In addition, the number of token queues in total is N^2, which can be a obstacle for the switch to scale up to hundreds or even thousands of ports.

In addition, the k-MC-VOQ architecture is proposed in [63]. Each input port maintains k FIFO queues, with $1 < k < 2^N - 1$. The main issues for the k-MC-VOQ architecture are related to the scheduling algorithm and the queuing discipline that associates each multicast flow with a queue. A Greedy Min-Split Scheduling (GMSS) [63] is proposed to schedule multicast traffic for the k-MC-VOQ architecture. Each queue is associated with a weight, which is the product of the queue length and the fan-out of the multicast cell at the head of the queue. Queues are examined by decreasing order of the weights. The scheduling algorithm iterates with two phases until either all output ports are selected or no more non-empty queues exist at unselected inputs. With an increase of k, i.e. the number of queues at each input, throughput improves only significantly for small k, i.e. $k \leq N$. Load balancing based on the queue length across multicast queues is required to distribute cells to different queues for performance improvement. This has made the system complex for implementation.

4.3 System Architecture and Problem Definition

The system architecture is presented in this section, followed by the problem definition. The notations in the system model are used throughout the rest of this chapter for consistency.

4.3.1 System Architecture

The multicast cell-based switching system used in this chapter is assumed to have the architecture described in Figure 4.6. The switch is assumed to have equal number of input and output ports due to the fact that an input and an output port usually reside in pair on the same line card. Each input i, $0 \leqslant i \leqslant N-1$, is connected to a FIFO queue. The information of cells is collected and scheduling decisions are made by the multicast scheduling module. The status of each output j, $0 \leqslant j \leqslant N-1$, is collected by the multicast scheduling module. We also assume that the switch fabric has intrinsic multicast/broadcast capabilities, e.g. crossbar switch fabric. Incoming variable-length packets are segmented into fixed-size cells before traversing the switch fabric and are reassembled at the output ports before being sent out. It is out of the scope of this chapter to consider the details and technologies of packet segmentation and reassembly. Thus we only focus on the multicast cell switching part. Sufficient buffer capacities are assumed so that no cell loss occurs due to the buffer overflow. Since variable-length packets are segmented into fixed-size cells, time is divided into fixed periods, denoted as *cell times*. In one cell time, an input can only send at most one cell to the switch fabric and an output can only receive at most one cell from the switch fabric. If more than one cells are bound for an output within a cell time, an output contention is occurred and only one cell can be scheduled for transmission according to the scheduling algorithm with other cells left to be scheduled in the next cell time.

Any multicast cell is characterized by its fan-out set, i.e. the set of the output ports for which the cell is bound. As a simple example shown in Figure 4.6, input 0 has a cell at the head of the queue destined to outputs {2, 3, 8}, and fan-out set can thus be expressed as {2, 3, 8}. We consider the case where *fan-out splitting* [76] is applied so that copies

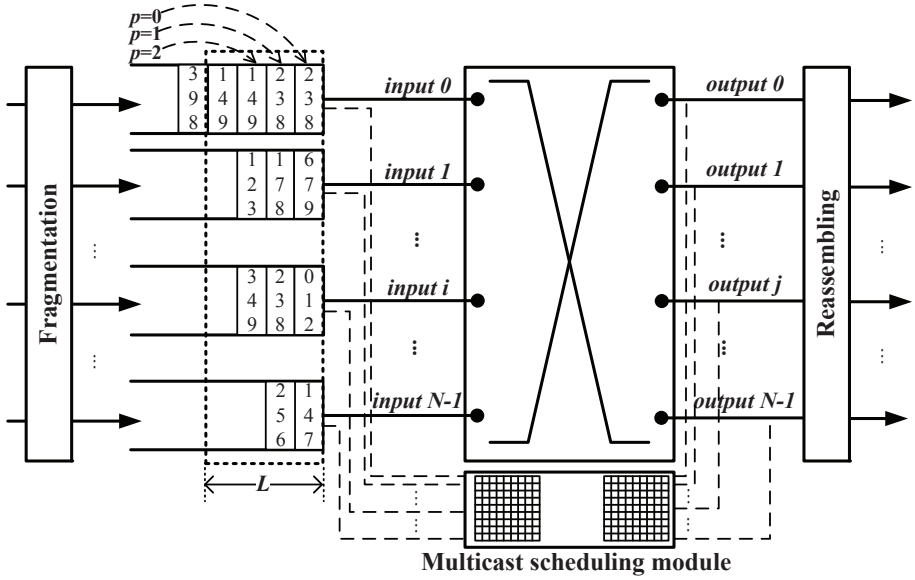


Figure 4.6: The system model of the multi-level round-robin multicast scheduling algorithm.

of multicast cells can be delivered to output ports over any number of cell times. Unless all the destinations in the fan-out set are reached, the cell will not be removed but remain in the queue. A multicast scheduler makes scheduling decisions prior to each cell time and grants cell transmissions accordingly.

4.3.2 Problem Definition

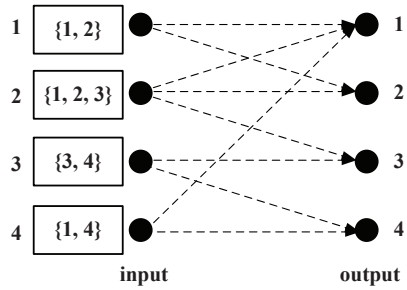
An efficient way to schedule multicast cells is to see the cells from the output's point of view. Even though a multicast cell is bound for several output ports, for a specific output port, it only takes into account whether the multicast cell is destined to itself. For example in Figure 4.7, the fan-out information of the HOL cells in each input queue is shown and a diagram can be created to represent all the fan-out information. For output 1, the fan-out information for output 2, 3, and 4 is useless and therefore a subdiagram can be created which filters out all the fan-out information for other output ports, as shown in Fig-

ure 4.7(b). Similarly, subdiagrams for output 2, 3, and 4 are shown in Figure 4.7(c), Figure 4.7(d), and Figure 4.7(e), respectively. Based on this way of scheduling, the round-robin scheduling algorithm can run independently on each output to select a cell for transmission. This guarantees that an output can always succeed in scheduling a cell to be sent to it, as long as the fan-out information of the cells includes the output.

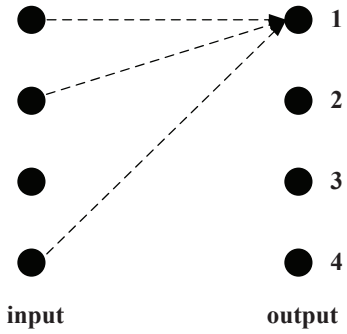
The MLRRMS algorithm is proposed based on this principle. However, two crucial problems should be solved. If the scheduling algorithm only operates on the HOL cells, the system can suffer from the HOL blocking problem, where the cell is blocked by the one ahead of it and loses its chance to be sent to the idle output ports. To illustrate this problem clearly, an example is shown in Figure 4.8. A multicast cell in the FIFO queue for input 1 is scheduled to be sent to output 1 and 2, for instance. The other two HOL cells from input 2 and 3 lose the chance to be sent and are to be scheduled in the next cell time. As a result, output 3 is idle for this cell time and the two cells in queue 1 and 3 are blocked from transmission. The throughput is thus reduced by the HOL blocking problem. For unicast traffic, the throughput of a IQ switch is limited to 58.64% by the HOL blocking problem [59].

The other problem is the number of transmissions of each multicast cell. As described, a multicast cell is removed from the queue *when and only when* all its destinations are reached. This implies that a multicast cell can be transmitted up to as many times as the fan-out value if the scheduling algorithm fails to utilize the multicast/broadcast capability of the switch fabric. Since each output port makes the scheduling decision independently, it is possible that they select different cells even if some can reach all the destinations within one cell time and be removed from the queues. This unnecessary multiple transmissions of multicast cells can result in an increased cell delay since the system takes more cell times to remove a multicast cell from the queue than a system with an advanced algorithm to avoid such a situation.

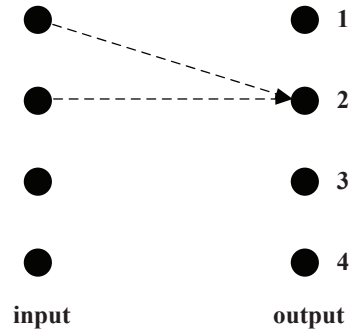
To alleviate the addressed problems, the Look-Ahead (LA) and the *sync* mechanism are proposed in the MLRRMS algorithm. The sync mechanism aims to reduce the unnecessary multiple transmissions of a multicast cell. The LA mechanism aims to reduce the HOL blocking and is used by the MLRRMS based on the assumption that the scheduler is



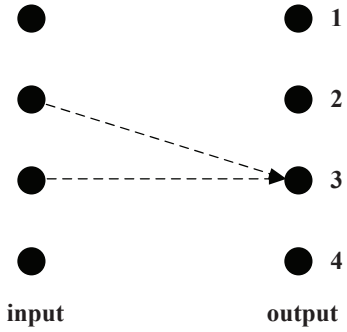
(a) The full diagram.



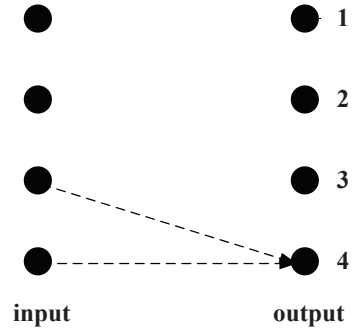
(b) The subdiagram for output 1.



(c) The subdiagram for output 2.



(d) The subdiagram for output 3.



(e) The subdiagram for output 4.

Figure 4.7: Illustration of splitting a multicast scheduling problem. For each output port, a diagram can be created which filters out all the fan-out information for other output ports.

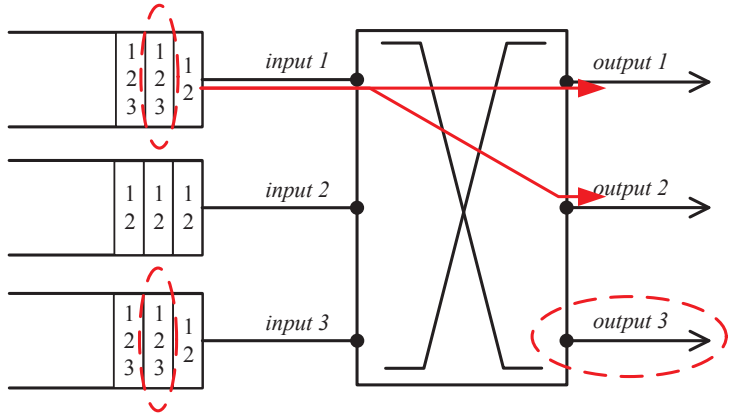


Figure 4.8: An example of the multicast head-of-line (HOL) blocking problem. Output 3 is idle and the two cells in queue 1 and 3 are blocked from transmission by the cell ahead.

able to examine the cells stored further in the queues and is capable of sending them to the corresponding output ports.

4.4 The Multi-Level Round-Robin Multicast Scheduling Algorithm

The MLRRMS algorithm is a distributed multicast scheduling algorithm, though it is assumed to be implemented in one module to reduce the signal latency. The terms *input* and *output* used in the algorithm description are not necessarily the actual inputs and output of the switch, but rather a conceptual indication for scheduling purposes. The MLRRMS can reduce the unnecessary multicast transmissions of a multicast cell by using the sync mechanism. Unlike the WBA, which operates only on the HOL cells, the MLRRMS uses the LA mechanism to iterate the scheduling process on different cell position to increase the throughput. The detailed description of the MLRRMS algorithm is shown as below and an example is shown in Figure 4.9 and Figure 4.10:

Initial condition: Before each cell time, the position pointer, p, is reset to point to the HOL cell, i.e. $p = 0$. All the input and output ports are in unreserved status and are eligible of transmitting and receiving

cells.

Step 1) Submission: Each unreserved input submits to the unreserved outputs which are contained in the fan-out set of the cell pointed by the position pointer p. If $p+1$ is larger than the queue length, the input stops this step. The output ports that have received the submissions from the inputs will appear in a round-robin schedule of the dictator assignment.

Step 2) Dictator Assignment: The dictator arbiter of the current position pointer chooses the output that appears next in a round-robin schedule, starting from the highest priority element, to be the dictator over other outputs. The dictator pointer $a^{(p)}$ to the highest priority element of the round-robin schedule is incremented (modulo N) to one position beyond the current dictator, after the assignment.

Step 3) Decision: If an unreserved output receives any fan-out information submissions, it chooses the one that appears next in a round-robin schedule of the current position pointer, starting from the highest priority element. The output notifies each input whether its submission is selected in the decision and becomes reserved. The decision pointer $d^{(p)}$ to the highest priority element of the round-robin schedule, is incremented (modulo N) to one location beyond the selected input, *if and only if*, the output receives a cell from its selected input. Upon receiving a decision, the input temporarily stores the index of the output that has sent this decision, as well as the value of the current position pointer.

Step 4) Sync: If an input receives a decision from the dictator of the current position pointer, it invalidates the decisions of other outputs, which are contained in its submission set, and makes its submissions in *Step 1* as valid decisions. An input without valid decisions loses permission to transmit cells and remain unreserved. Only an input having at least one valid decision becomes reserved and it is eligible for transmission.

Step 5) Look-Ahead: If any unreserved output port exists and if the position pointer has *not* reached its maximum value, the position pointer increases its value by 1, i.e. $p = p + 1$, go to *Step 1*. Else if all output ports are reserved or the position pointer has reached its maximum value, the scheduling process is completed.

After the completion of the scheduling process, each reserved input copies the cell in the FIFO queue from the position that has been stored

in *Step 3)* and sends cells to the outputs that are included in the decision set. If a cell has reached all the output ports in its fan-out set, it is removed from the queue. Otherwise, the cell remains in the queues, removes those reached outputs from its fan-out set and updates it fan-out information.

4.5 MLRRMS Algorithm Analysis

The analytical analysis of the MLRRMS algorithm is presented in this section. First, several terms are defined in Section 4.5.1. Then in Section 4.5.2, the analytical description of the MLRRMS algorithm is provided for the purpose of further analysis. Heuristic analysis of the LA mechanism is given in Section 4.5.3, and finally the complexity analysis is presented in Section 4.5.4.

4.5.1 Definitions

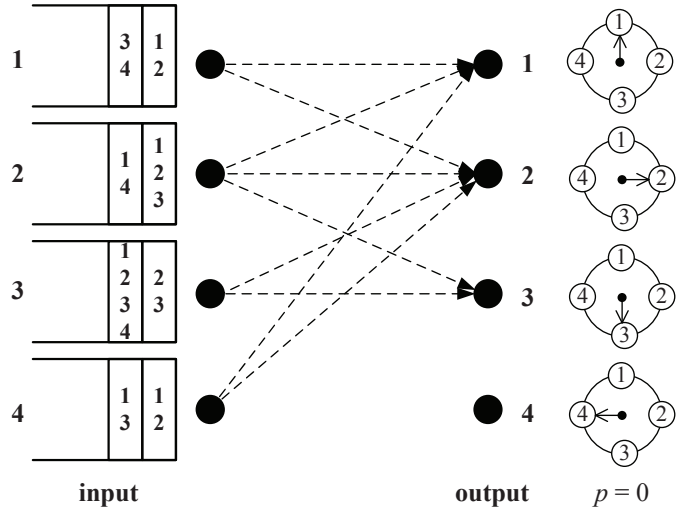
We define several terms used in the analysis of the MLRRMS algorithm:

Definition 1 (Maximum Look-Ahead Depth): The *maximum look-ahead depth*, L, is defined as the limit of the number of cells that the scheduler is able to examine further into the queue. $L = 0$ means that the switch only operates on the HOL cells, while $L = l$ indicates that the switch can look up to l cells after the HOL cell.

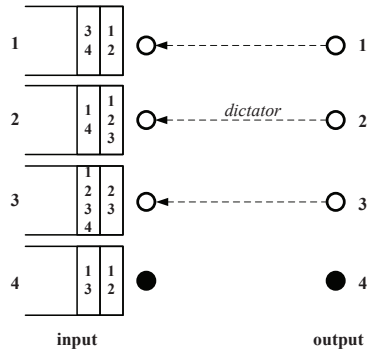
Definition 2 (Cell Position): The *cell position*, p, is defined as the position of a cell in the queue. The cell at the HOL of the queue has $p = 0$.

Definition 3 (Fan-out Vector): A *fan-out vector* is used to indicate the fan-out set carried by a multicast cell in input i at position p, and is denoted as $\mathbf{f}^{(i,p)} \triangleq f_k^{(i,p)}$, $k = 0, 1, ..., N-1$, $p = 0, 1, ..., L$, $f_k^{(i,p)} \in \{0, 1\}$. $f_k^{(i,p)} = 0$ indicates that output k is not in the fan-out set of the cell and $f_k^{(i,p)} = 1$ indicates the opposite. The cardinality of the fan-out set thus becomes $|\mathbf{f}^{(i,p)}| \triangleq \sum_{k=0}^{N-1} f_k^{(i,p)}$.

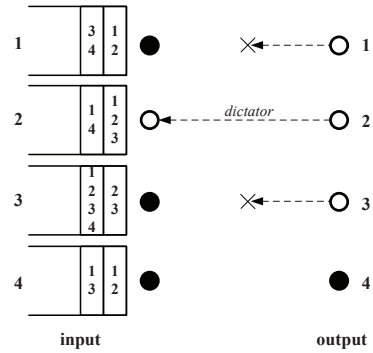
Definition 4 (Traffic Matrix): The *Traffic Matrix* is an $N \times N$ matrix constructed by the scheduler, based on the fan-out vectors of the cells in the position p of each input i, before a cell transmission. It is denoted as $\mathbf{T}^{(p)} = \left(T_{i,j}^{(p)}\right)$. Obviously, we have $T_{i,j}^{(p)} = f_{i,j}^{(p)}, \forall i, j, p$. We define



(a) Step 1 with $p = 0$: Submission. Each unreserved input submits the fan-out information of its HOL cell to the corresponding outputs. The round-robin scheduler ($p = 0$) of each output is at the position left from the last scheduling process.

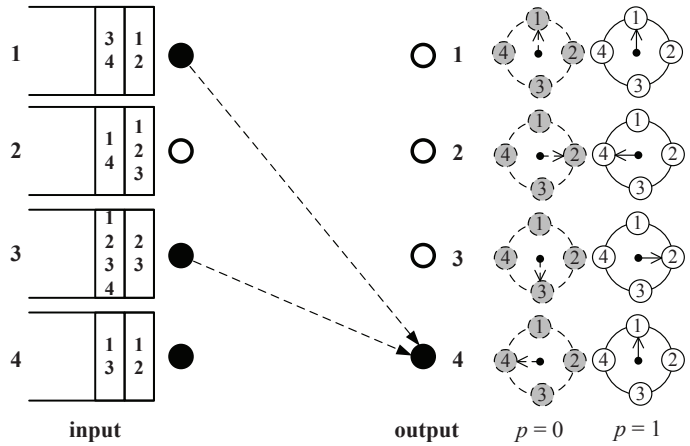


(b) Step 2 and 3: Dictator Assignment and Decision. Output 2 is the dictator in this round. Based on the round-robin pointer, each output sends a decision to an input and becomes reserved.

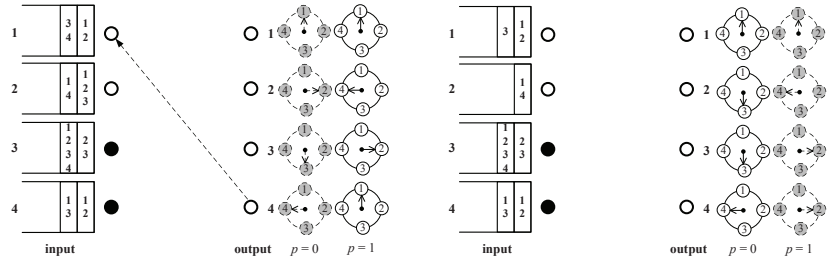


(c) Step 4: Sync. Input 2 receives a decision of the dictator and thus it invalidates the decisions sent by output 1 and 3, because they are in its fan-out set. Input 1 and 3 lose their decisions and therefore become unreserved.

Figure 4.9: MLRRMS: Submission, Decision, and Sync.



(a) Step 5 and Step 1 with $p = 1$: Look-Ahead and submission of the increased cell position. Since output 4 is unreserved, input 1 and 3 both submit the fan-out information of the cell at $p = 1$.



(b) Step 2 with $p = 1$: Decision with $p = 1$. Output 4 sends a decision to input 1 according to its round-robin pointer at $p = 1$, and becomes reserved.

(c) Post-transmission. The HOL cell in input 2 is sent to all its destinations and is removed from the FIFO queue. Since the cells received by output 1 and 3 are different from what the outputs' round-robin pointers indicate, no update occurs on the pointers.

Figure 4.10: MLRRMS: Look-ahead, Submission, Decision, and post-transmission status.

$T_{i,j}^{(p)} = 0, \forall j, p$, if input queue i is empty.

Definition 5 (Decision Matrix): The *Decision Matrix* is an $N \times N$ matrix denoted as $\mathbf{D}^{(p)} = \left(D_{i,j}^{(p)}\right)$, $D_{i,j}^{(p)} \in \{0, 1\}$. This matrix contains the scheduling decisions for each output j, with $D_{i,j}^{(p)} = 1$ indicating that a copy of the cell in input i at position p will be transferred to output j and $D_{i,j}^{(p)} = 0$ meaning that no copy will be sent to output j. Thus, $0 \leq \sum_j D_{i,j}^{(p)} \leq 1$, $\forall j$. $\mathbf{D}^{(p)}$ satisfies the conditions in Equation 4.1 4.2, and 4.3 as below:

$$0 \leq \sum_{j=0}^{N-1} D_{i,j}^{(p)} \leq N, \quad \forall i, p \tag{4.1}$$

$$0 \leq \sum_{i=0}^{N-1} D_{i,j}^{(p)} \leq 1, \quad \forall j, p \tag{4.2}$$

$$0 \leq \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} D_{i,j}^{(p)} \leq N, \quad \forall p \tag{4.3}$$

Definition 6 (Set of Decision Matrices): The *Set of Decision Matrices* is defined as $\mathbf{\Delta} \triangleq \{\mathbf{D}^{(0)}, \mathbf{D}^{(1)}, ..., \mathbf{D}^{(L)}\}$. It contains up to L decision matrices. Multicast cells are released by the scheduler according to the decision matrices stored in Δ.

Definition 7 (Assistant Matrix): The *Assistant Matrix* is an $N \times N$ matrix denoted as $\mathbf{A}^{(p)} = \left(A_{i,j}^{(p)}\right)$, $A_{i,j}^{(p)} \in \{0, 1\}$. This matrix is used to help generate $\mathbf{D}^{(p)}, p > 0$.

Definition 8 (Cross Disable Mark $\mathbf{X}^{\circ}$): We define $\mathbf{X}^{\circ}$ as a matrix transform mark for the sake of convenience, where $\mathbf{X} = (X_{i,j})$, $X_{i,j} \in 0, 1$ is the matrix in operation. If we have $\mathbf{Y} = \mathbf{X}^{\circ}$, first let $\mathbf{Y} = \mathbf{O}$, ($Y_{i,j} = 0$, $\forall i, j$) with the same dimensions as $\mathbf{X}$, and if $X_{k,l} = 1$, then $Y_{k,j} = 1$, $Y_{i,l} = 1$, $\forall i, j$.

4.5.2 Analytical Description of the MLRRMS Algorithm

We here describe the proposed multicast scheduling algorithm in detail based on the previous definitions. Before each cell transmission time,

the scheduler executes the following procedures and accordingly releases cells after completion.

Initial condition: $p = 0$, $\mathbf{\Delta} = \emptyset$, and $\mathbf{D}^{(-1)} = \mathbf{O}$ ($D_{i,j}^{(-1)} = 0, \forall i, j$).

i): The scheduler examines the fan-out vector $\mathbf{f}^{(i,p)}$ of the cell in input i at position p for all inputs to construct $\mathbf{T}^{(p)}$.

ii): $\mathbf{A}^{(p)} = \mathbf{T}^{(p)} - \left(\sum_{p=0}^{|\mathbf{\Delta}|} \mathbf{D}^{(p)}\right)^{\circ}$, and if $A_{i,j}^{(p)} < 0$, then set $A_{i,j}^{(p)}$ to 0, $\forall i, j$.

iii): The round-robin scheduling algorithm is independently executed on each non-zero column of $A^{(p)}$. Only one element in a column can be selected due to the constraint of one output port only being able to one transmission during a cell time. The scheduling results thus form $D^{(p)}$.

iv): The sync procedure is carried out on $\mathbf{D}^{(p)}$ to reduce the unnecessary multiple transmissions of cells caused by the independent scheduling processes: each column plays the role of dictator in a round-robin manner, and if column y plays the role of dictator during the current cell time and $D_{x,y}^{(p)} = 1$, and $\forall j \neq y$, $A_{x,j}^{(p)} = 1$ and $D_{x,j}^{(p)} \neq 1$, then let $D_{x,j}^{(p)} = 1$ and $D_{x,j}^{(p)} = 0$, $\forall i \neq x$. The scheduler stores the refined $D^{(p)}$to $\mathbf{\Delta}$, i.e. $\mathbf{D}^{(p)} \rightarrow \mathbf{\Delta}$. The round-robin pointer of the column that is synced to the dictator remains in the same position for the next cell time.

v): If a zero column is found in $\sum_{p=0}^{|\mathbf{\Delta}|} \mathbf{D}^{(p-1)}$, check the queue size of each unreserved input, which is the corresponding row in $\sum_{p=0}^{|\mathbf{\Delta}|} \mathbf{D}^{(p-1)}$. If the queue size is larger than $p + 1$, and $p + 1 \leq L$, increase p with 1 and go to step *i*. Otherwise, continue to step *vi*.

vi): The scheduler should examine $\mathbf{\Delta}$ and release multicast cells at particular positions from input queues according to each $\mathbf{D}^{(p)}$. If the fan-out set of a cell becomes empty after the service, the cell will be removed from the queue. Otherwise, the cell remains with a new fan-out set.

4.5.3 Heuristic Analysis of the Look-Ahead Mechanism

As described previously in the algorithm, the LA mechanism is only performed when the output ports are not fully reserved, which can cause decreased throughput. There are potentially two reasons to cause the partial output port occupancy: (1) the HOL blocking, and (2) the traffic pattern. Obviously, if the reason of the partial output port occupancy

is the traffic pattern, there is nothing to improve. On the other hand, the HOL blocking phenomenon may be the cause and therefore the LA mechanism is introduced to reduce the problem.

It is obvious that the HOL blocking can be eliminated if the switch is capable of searching infinitely in the queues, i.e. the maximum LA depth is always larger than the queue size. However, taking the implementation complexity into consideration, infinite searching capability is impractical. As defined previously, a maximum LA depth is introduced to constrain the implementation complexity and at the same time, to increase the output utilization. Here in this section, a discussion is carried out on the relation between the maximum LA depth and the output utilization.

Assuming that there are enough cells stored in the queues and the fan-out vectors are uniformly distributed among cells, i.e. a multicast cell is bound for each output port with the same probability:

$$P\left(f_k^{(i,p)} = 1\right) = \delta, \quad \forall i, k, p \tag{4.4}$$

A multicast cell always carries a non-zero fan-out vector, i.e. there is at least one destination in the fan-out set if unicast is considered as a special case of multicast, the probability of the fan-out can be calculated. A random variable $F = |\mathbf{f}^{(i,p)}|$ is defined for the fan-out of a multicast cell, and the probability of $F = f$ is calculated as:

$$P(F = f) = \frac{\binom{N}{f}\delta^f(1-\delta)^{(N-f)}}{1-(1-\delta)^N}, \quad f = 1, 2, ..., N \tag{4.5}$$

$$E[F] = \frac{N \cdot \delta}{1-(1-\delta)^N} \tag{4.6}$$

Given the restriction in Equation 4.5, it is possible to derive the probability of an random element in $\mathbf{T}^{(p)}$ being 1:

$$\begin{aligned} P\left(T_{i,j}^{(p)} = 1\right) &= \frac{\delta}{1-(1-\delta)^N} \cdot \left(\frac{N}{N}\right) \\ &= \frac{\delta}{1-(1-\delta)^N} \\ &= \theta_1^{(p)} \end{aligned} \tag{4.7}$$

Therefore the probability of *1* random column in $\mathbf{T}^{(0)}$ being zero is:

$$\varphi_1^{(0)} = P\left(\sum_j T_{i,j}^{(0)} = 0\right) = \left(1 - \theta_1^{(0)}\right)^N, \quad \forall i, j \tag{4.8}$$

Given one zero column, the probability of a random element in the rest $N-1$ columns of $\mathbf{T}^{(0)}$ being 1 given one zero column is known as:

$$\theta_2^{(0)} = \frac{\delta}{1-(1-\delta)^N} \cdot \left(\frac{N}{N-1}\right) \tag{4.9}$$

The probability of a second random column in $\mathbf{T}^{(0)}$ being zero given one zero column is known is:

$$\varphi_2^{(0)} = \left(1 - \theta_2^{(0)}\right)^N \tag{4.10}$$

Thus, the probability of *2* random columns in $\mathbf{T}^{(0)}$ being zero is:

$$P\left(\textit{2} \text{ random zero colums in } \mathbf{T}^{(0)}\right) = \varphi_1^{(0)} \cdot \varphi_2^{(0)} \tag{4.11}$$

Suppose that there are x zero columns in $\mathbf{T}^{(0)}$, and we can derive:

$$\theta_x^{(0)} = \frac{\delta}{1-(1-\delta)^N} \cdot \left(\frac{N}{N-x+1}\right) \tag{4.12}$$

$$\varphi_x^{(0)} = \left(1 - \theta_x^{(0)}\right)^N \tag{4.13}$$

Thus, define a random variable $\mathbf{X}^{(0)}$ is defined for the number of zero columns in $\mathbf{T}^{(0)}$ and the probability of $\mathbf{X}^{(0)} = x$ is calculated as:

$$P\left(X^{(0)} = x\right) = \begin{cases} \binom{N}{x}\left(\varphi_1^{(0)}\varphi_2^{(0)}\cdots\varphi_x^{(0)}\right)\left(1-\varphi_x^{(0)}\right)^{N-x}, 1 \le x \le N-1 \\ \left(1-\varphi_1^{(0)}\right)^N, \quad x = 0 \end{cases} \tag{4.14}$$

If zero columns exist in $\mathbf{T}^{(0)}$, they are further examined in $\mathbf{T}^{(p)}, p > 0$, and assume that each bit in a non-zero column has the equal chance of being selected by the round-robin scheduler. Then it is possible to derive:

$$P(\text{a random bit in a non-zero column is selected}) = \frac{\theta_x}{N} \tag{4.15}$$

If the scheduling decisions on those columns are fully scattered on different $N - x$ rows, then $N - x$ rows will be disabled in $\mathbf{T}^{(p)}, p > 0$. The probability of this situation is:

$$\begin{aligned} P(\text{fully-scattered}) &= \binom{N}{1}\frac{\theta_x}{N} \cdot \binom{N-1}{1}\frac{\theta_x}{N} \cdots \binom{x+1}{1}\frac{\theta_x}{N} \\ &= \left(\frac{\theta_x}{N}\right)^{N-x} \cdot \prod_{b=1}^{N-x}\binom{N-b+1}{1} \end{aligned} \tag{4.16}$$

If the scheduling decisions on those columns are all the same row, only 1 row will be disabled for further look-ahead process. The probability of this situation is:

$$\begin{aligned} P(\text{zero-scattered}) &= \binom{N}{1} \cdot \left(\frac{\theta_x}{N}\right)^{N-x} \\ &= \left(\frac{\theta_x}{N}\right)^{N-x} \cdot \prod_{b=1}^{1}\binom{N-b+1}{1} \end{aligned} \tag{4.17}$$

In between the above two extreme cases described in Equation 4.16 and Equation 4.17, the probability that the scheduling decisions are scattered among α different rows, where $1 < \alpha < N - x$, can be calculated:

$$\begin{aligned} P(\alpha\text{-scattered}) &= \binom{N}{1}\left(\frac{\theta_x}{N}\right)^{N-x-\alpha+1} \cdot \\ &\quad \binom{N-1}{1}\frac{\theta_x}{N} \cdot \binom{N-2}{1}\frac{\theta_x}{N} \cdots \binom{N-\alpha+1}{1}\frac{\theta_x}{N} \\ &= \left(\frac{\theta_x}{N}\right)^{N-x} \cdot \prod_{b=1}^{\alpha}\binom{N-b+1}{1}, \quad 1 < \alpha < N - x \end{aligned} \tag{4.18}$$

Thus, combining the three situations, one generic formula is derived for the probability of α rows being disabled in $\mathbf{T}^{(1)}$ as shown in Equation 4.19:

$$P_d(\alpha) = \left(\frac{\theta_x}{N}\right)^{N-x} \cdot \prod_{b=1}^{\alpha} \binom{N-b+1}{1}, \quad 1 \leq \alpha \leq N-x \tag{4.19}$$

Then the probability of *1* random column in $\mathbf{T}^{(1)}$ being zero can be calculated:

$$\varphi_1^{(1)} = P_d(\alpha) \left(1 - \theta_1^{(1)}\right)^{N-\alpha} \tag{4.20}$$

where $\theta_1^{(1)} = \theta_1^{(0)}$ since each $\mathbf{T}^{(p)}, \forall p$ is independent.

Therefore the probability of no zero columns existing after looking *1* cell behind the HOL cell becomes:

$$P\left(X^{(0)} = x\right) P\left(X^{(1)} = 0\right) = \binom{N}{x} \left(\varphi_1^{(0)} \varphi_2^{(0)} \cdots \varphi_x^{(0)}\right) \left(1 - \varphi_x^{(0)}\right)^{N-x} \cdot \left(1 - \varphi_1^{(1)}\right)^x \tag{4.21}$$

Given a large N and $\delta = 0.5$, the probability shown in Equation 4.21 can be high, which will be later demonstrated by simulation results in Section 4.6. This indicates that by allowing the switch to look 1 cell further into the queues after the HOL cell, the largest improvement on increasing the output utilization can be achieved.

4.5.4 Complexity Analysis

The time complexity of each step of the MLRRMS algorithm is discussed in this section.

First of all, the time complexity of *Submission* for each input can be $O(N)$ because the input needs to examine all N bits of the fan-out vector carried by the cell. If parallel structure is used to allow the input to read the N bits at one access, the time complexity can be reduced to $O(1)$.

Secondly, for each *Decision* arbiter to generate a decision, the time complexity can be $O(N)$ because the scheduling arbiter needs to examine at most N submissions before sending the decision back to an input. However, if we use priority encoders, the time complexity can be reduced to $O(\log N)$. The *Dictator Assignment* arbiter can also use priority encoder to reduce the time complexity of the dictator output from $O(N)$ to $O(\log N)$. The sync mechanism is simple to implement and has a time complexity of $O(1)$ since the input that receives the decision from the dictator only needs to invalidate the decisions from the outputs included in the submission set.

Give a maximum LA depth, L, the scheduling process iterates until either all outputs are reserved or the maximum LA depth is reached. Thus, the time complexity becomes $O(L \log N)$.

4.6 Simulated Performance of MLRRMS

A comparison between the MLRRMS, the WBA [62] and the FIFOMS [60] is carried out by simulations in OPNET Modeler [56]. Independent multicast traffic is assumed to each input. To compare the performance of the algorithms in varying traffic conditions, Bernoulli traffic and bursty traffic with different fan-out schemes are considered, which are further explained in Section 4.6.1.

4.6.1 Traffic Model

For Bernoulli traffic process, a cell arrives at an input with a probability of q, which is also the arrival rate. Thus the offered load can be calculated as $\lambda = q \cdot E[F]$, where F is the random variable of the fan-out of each cell.

The bursty traffic process, or *Correlated Arrival Process*, has two states, *busy* and *idle*. Cells are generated only in the *busy* state. The process stays in each state for a random number of cell times following the geometric distribution with mean values of $E[B]$ and $E[I]$, respectively. The arrival rate is calculated as $q = E[B]/(E[B] + E[I])$, and the offered load can be calculated as $\lambda = q \cdot E[F]$. Since the traffic arrives at the switch in bursts, two modes of fan-out schemes can be applied, cell-based and burst-based. In cell-based fan-out mode, the fan-out vector

is independently generated for each cell. And in burst-based mode, the fan-out vector is independently generated for each burst of cells, each burst of cells having the same fan-out vectors.

To evaluate the multicast scheduling algorithms, the multicast balanceness is introduced. We can define that the probability of a bit in a fan-out vector of a cell in an input being 1 follows the Equation 4.22:

$$P\left(f_k^{(i,p)}=1\right)=\begin{cases}\max\left\{1, E[F]\cdot\left(\omega+\frac{1-\omega}{N}\right)\right\}, & k=i,\forall p\\ E[F]\cdot\left(\frac{1-\omega}{N}\right), & k\neq i,\forall p\end{cases} \tag{4.22}$$

where ω is the balance factor, $0 \leq \omega \leq 1$. When $\omega = 0$, all the bits have the same probability of being 1 in the fan-out vector, which indicates a balanced fan-out. As ω increases, the bit that has the same index as the input has higher probability of being 1. When ω reaches 1, the traffic becomes unicast, which can be considered an extremely unbalanced multicast case.

4.6.2 Performance for Balanced Multicast Traffic under Different Offered Loads

When $\omega = 0$, the multicast traffic becomes balanced and each bit in the fan-out vector has the same probability of being 1, resulting in a binomial distribution. Bernoulli traffic is first applied to the switch with $N = 8$, $E[F] = 4$ and $N = 32$, $E[F] = 16$. The reason that mean fan-out is half of the number of output ports is based on the assumption that, the probability of one multicast cell being sent to an output port is 0.5.

Figure 4.11 compares the average multicast delays under different traffic loads. A multicast cell is stored in the queue until all the destinations in its fan-out set are reached. The multicast delay of a cell is calculated as the cell times that the cell stays in the queue until it is removed. Since the WBA and the MLRRMS ($L = 0$) both operate only on the HOL cells, they become unstable under high offered loads. With looking ahead maximum 1 cell further, the MLRRMS ($L = 1$) has demonstrated a significant improvement of the multicast delay compared to the MLRRMS ($L = 0$) and the WBA. As L increases, Figure 4.11 displays more improvement from the MLRRMS ($L = 2$) and ($L = 10$), but the marginal improvement is decreasing. That is, the improvement

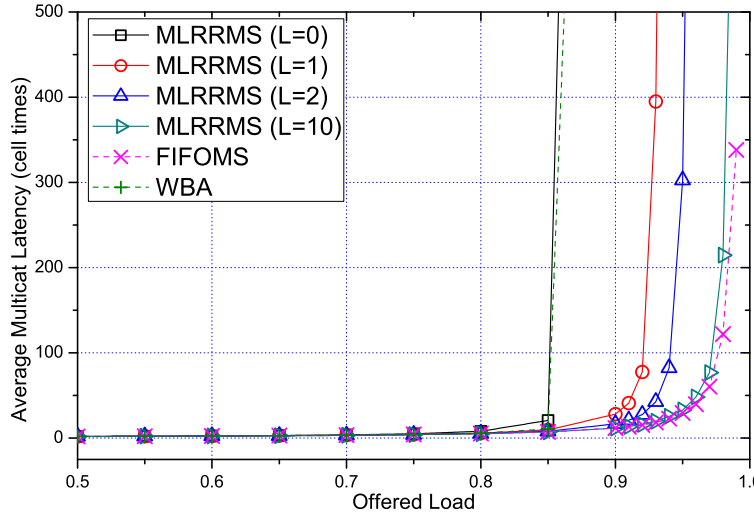
is not in proportion to the hardware implementation complexity added to the switching system to enable the switch to search more cells. From both Figure 4.11(a) and Figure 4.11(b), it is possible to point out that by allowing the switch system to be capable of looking 1 cell stored further in the queues, the system is able to obtain the largest improvement, in terms of the multicast latency. Among all, the FIFOMS has the lowest delay because it uses the VOQ architecture to handle the multicast traffic with a total number of queues of N^2, but as discussed previously, the VOQ architecture is low in scalability.

The average queue size per input, including the cell in service, is examined in Figure 4.12. Since the MLRRMS ($L = 0$) and the WBA operate only on the HOL cells, they both suffer from the HOL blocking problem and have the highest average queue size compared to other schemes. Again, a significant improvement of the MLRRMS ($L = 1$) in both Figure 4.12(a) and Figure 4.12(b) can be observed.

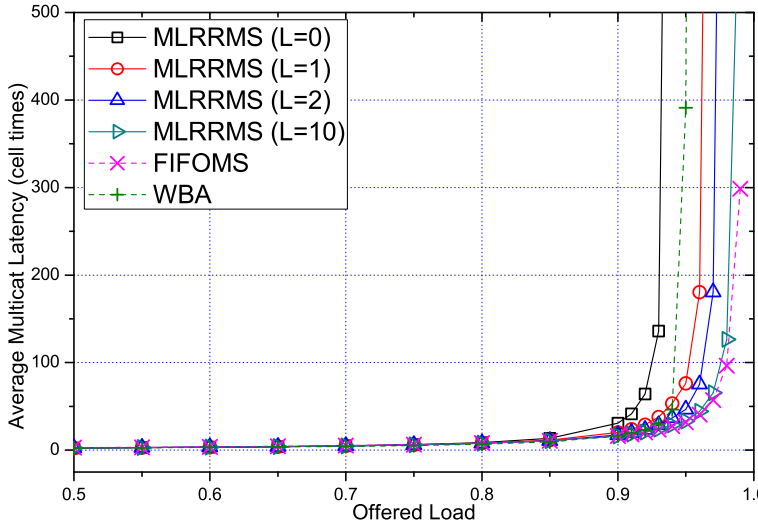
In Figure 4.13, the average LA depth is evaluated for both $N = 8$ and $N = 32$, i.e. the actual number of cells searched by the switch. For the MLRRMS ($L = 0$), the average LA depth is obviously always 0. For the MLRRMS ($L = 1$), it allows the switch to search up to 1 cell further in the queues. When the traffic load is heavy, the average LA depth is almost the same as L, which is set to 1. For the MLRRMS ($L = 2$) and ($L = 10$), the average LA depth under heavy load is less than its L value revealing that the switch does not utilize its full potential. The average LA depth of the MLRRMS ($L = 10$) is approximately 7 when the switch is heavily loaded with $N = 8$ and 8 with $N = 32$, indicating that the added implementation complexity of the switch is obsolete and the performance improvement is nonlinear. Both MLRRMS ($L = 1$) and ($L = 2$) begin to converge after $\lambda = 0.9$ because the queue size begin to become larger than the L values.

Bursty traffic is further applied with $N = 8, E[F] = 4$ and $N = 32, E[F] = 16$ and the mean burst size $E[B] = 16$ [62].

Performance comparisons are first carried out under the bursty traffic with the cell-based fan-out mode. In Figure 4.14, the average multicast latency of all the scheduling schemes increases. The WBA and the MLRRMS ($L = 0$) have the largest delay compared to others. With looking up to 1 cell, the MLRRMS ($L = 1$) has reduced the multicast latency dramatically. The MLRRMS ($L = 2$) does not provide the same

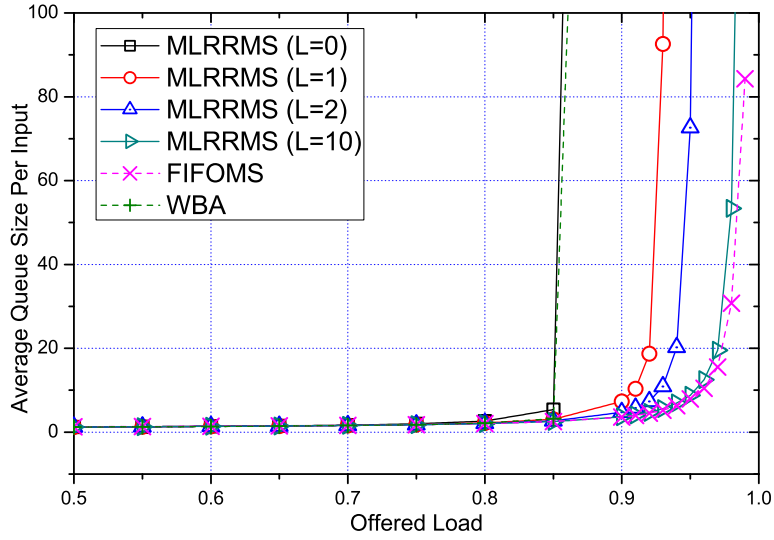


(a) $\omega = 0$, $N = 8$, $\bar{F} = 4$

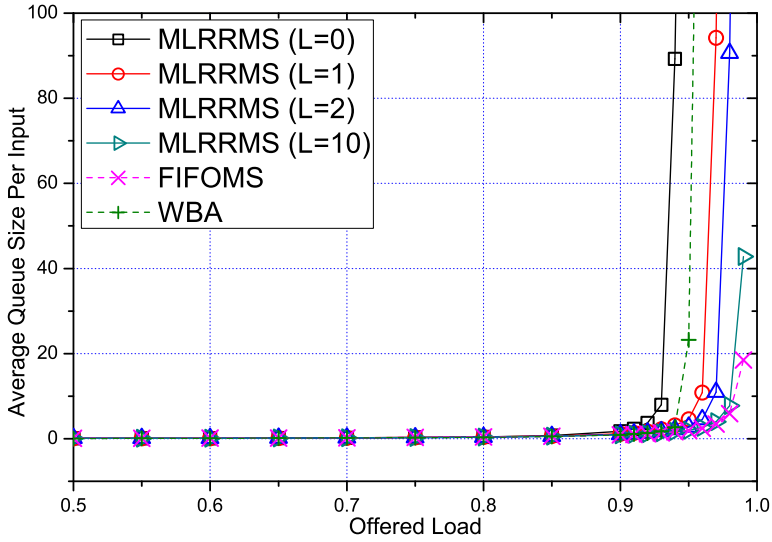


(b) $\omega = 0$, $N = 32$, $\bar{F} = 16$

Figure 4.11: Average multicast latency under Bernoulli traffic.

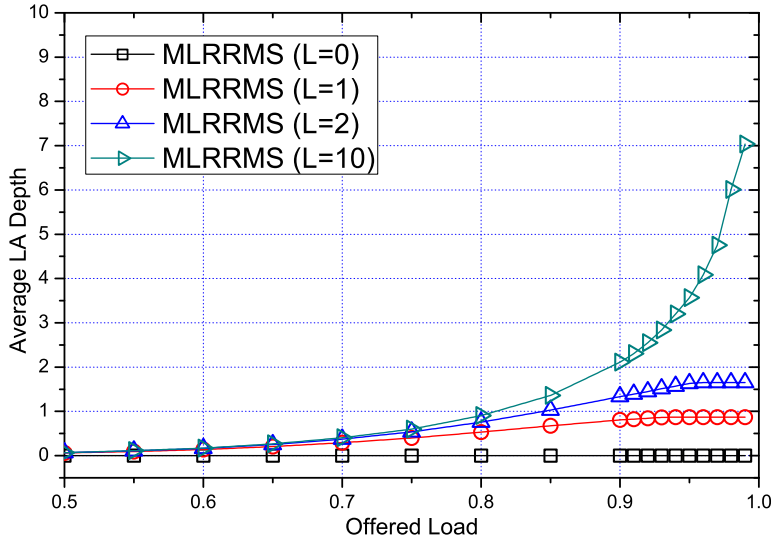


(a) $\omega = 0$, $N = 8$, $\bar{F} = 4$

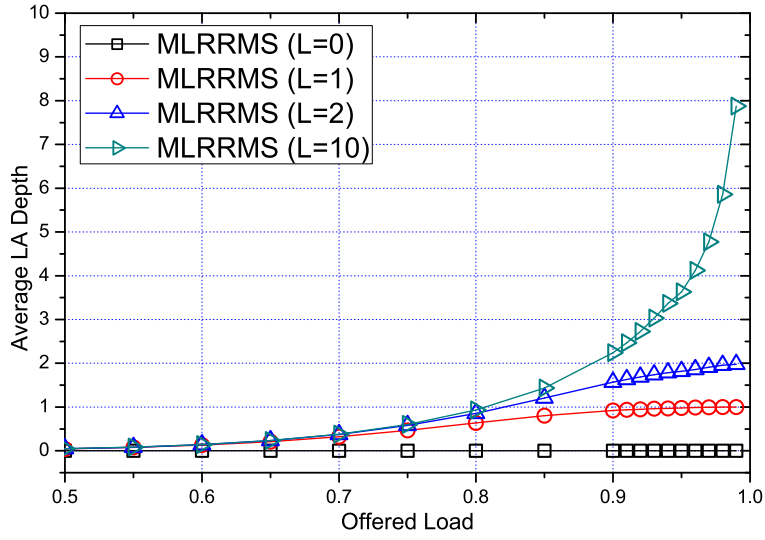


(b) $\omega = 0$, $N = 32$, $\bar{F} = 16$

Figure 4.12: Average queue size per input under Bernoulli traffic.



(a) $\omega = 0$, $N = 8$, $\bar{F} = 4$



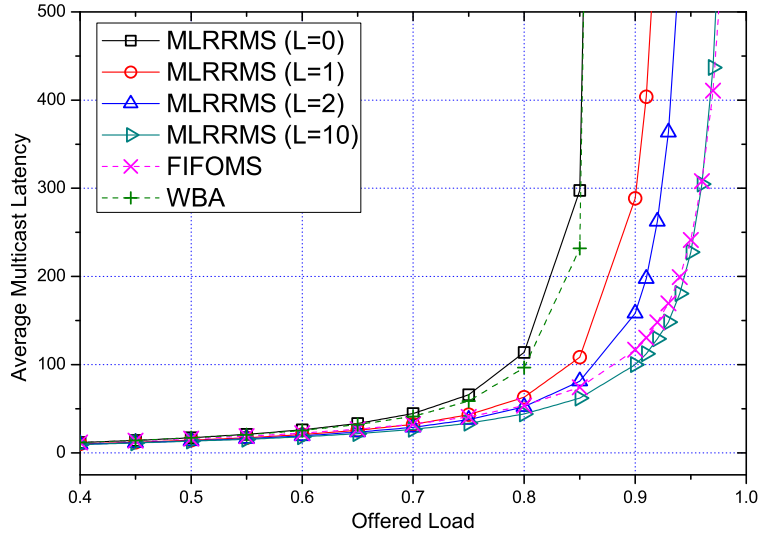
(b) $\omega = 0$, $N = 32$, $\bar{F} = 16$

Figure 4.13: Average look-ahead depth under Bernoulli traffic.

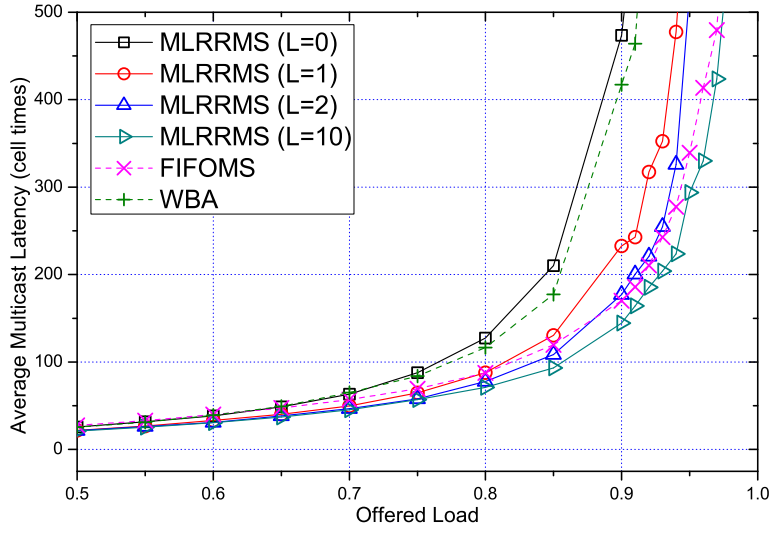
level of improvement compared to the complexity it adds to the switch. The delay performances of the MLRRMS ($L = 10$) and the FIFOMS are nearly the same under heavy traffic loads. In Figure 4.15, the average queue size per input is examined. Due to the bursty characteristic of the traffic, the average queue size is larger for light traffic load than when the Bernoulli traffic is applied. Since the fan-out scheme is cell-based, the LA mechanism can still reduce the HOL blocking problem and increase the output utilization of the switch. This can be observed in the improvement of the MLRRMS ($L = 1$) in both Figure 4.14 and Figure 4.15. In Figure 4.16, similar results are perceived as in Figure 4.13 but the average LA depth of the MLRRMS ($L = 2$) converges earlier than in Figure 4.13. This is because the bursty traffic leads to larger queue size under light load than the Bernoulli traffic.

Performance comparisons are then carried out under the bursty traffic with the burst-based fan-out mode. Each burst of cells has the same fan-out vector, which means looking ahead a limited number of cells in the queues is enough for the switch to alleviate the HOL blocking problem. A deeper searching should be carried out for this type of traffic. Among other schemes, the WBA has the largest multicast latency as shown in Figure 4.17. The FIFOMS performs better than the MLRRMS with different L values. In the MLRRMS group, the improvement of the MLRRMS ($L = 1$) is not as significant as in Figure 4.11 and Figure 4.14, and the MLRRMS ($L = 0$), ($L = 1$) and ($L = 2$) have similar multicast delay. This phenomenon corresponds to the reason stated previously, i.e. the burst-based mode requires a larger L, since $L = 1$ or 2 is not enough to include most possibilities of the burst distribution. The MLRRMS ($L = 10$) is able to generate a reduce the latency significantly because of the large L.

In Figure 4.18, we obtain similar results as in Figure 4.17. All algorithms suffer from performance degradation due to the traffic pattern. The average LA depths of the MLRRMS with different L's under burst-based mode are shown in Figure 4.19. The MLRRMS ($L = 1$) and ($L = 2$) begin to converge when the offered load λ reaches 0.6 due to the traffic pattern. The MLRRMS ($L = 10$) also converges at $\lambda = 0.8$ which indicates that the traffic of burst-based fan-out mode has put a greater demand on the possible LA depth than the cell-based scheme and the queue size is always larger than 10.

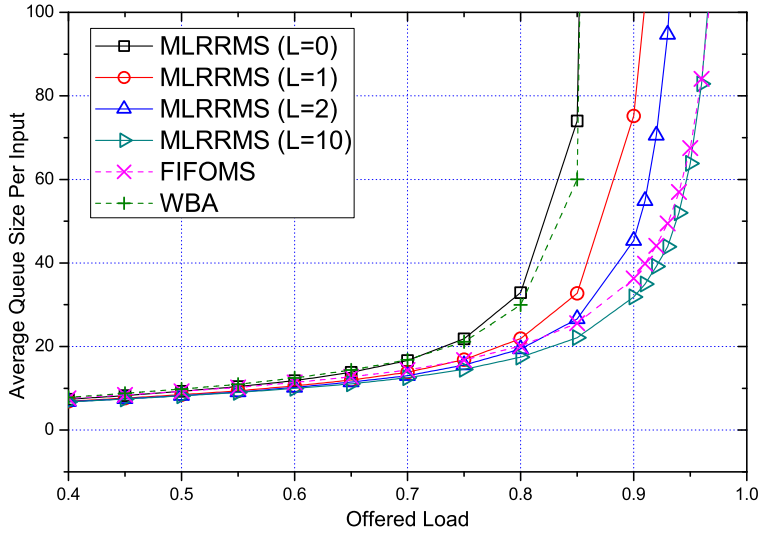


(a) $\omega = 0$, $N = 8$, $\bar{F} = 4$

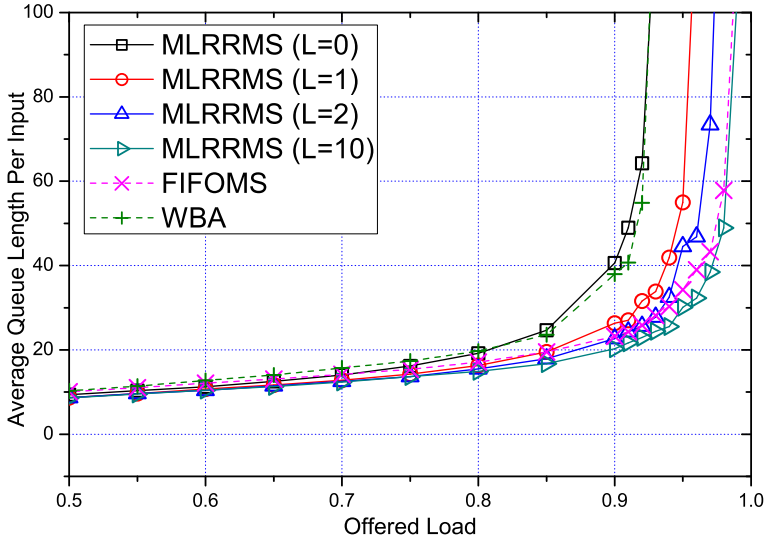


(b) $\omega = 0$, $N = 32$, $\bar{F} = 16$

Figure 4.14: Average multicast latency under bursty traffic (cell-based fan-out mode).

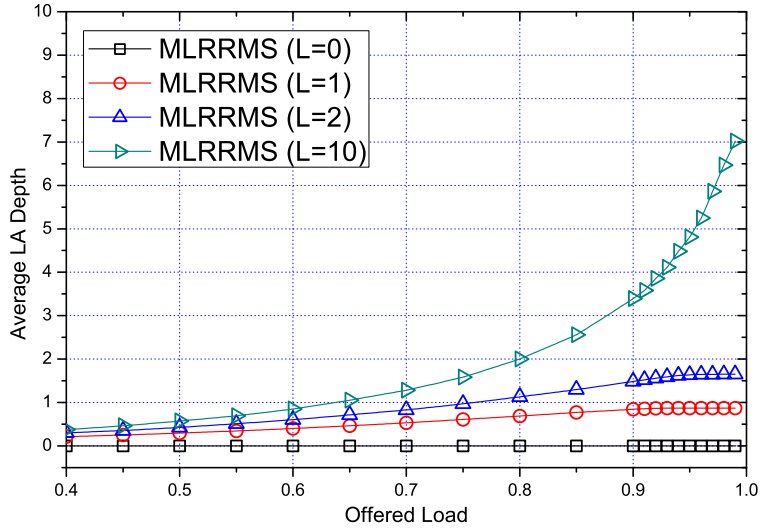


(a) $\omega = 0$, $N = 8$, $\bar{F} = 4$

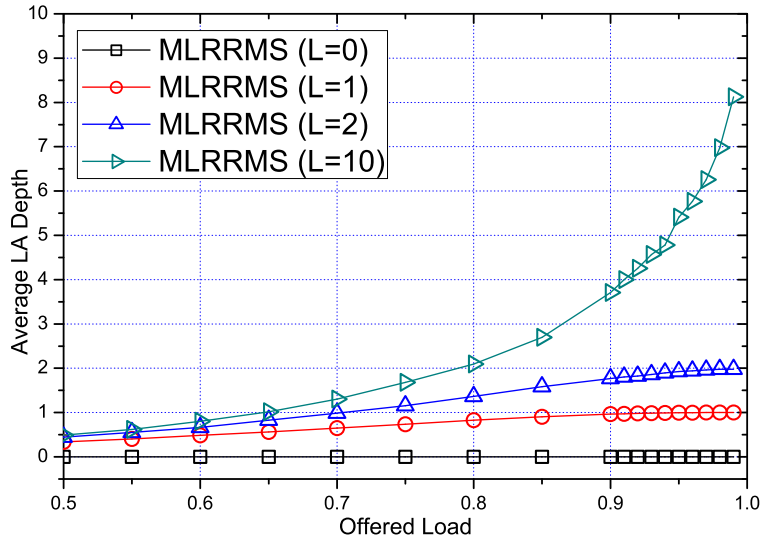


(b) $\omega = 0$, $N = 32$, $\bar{F} = 16$

Figure 4.15: Average queue size per input under bursty traffic (cell-based fan-out mode).

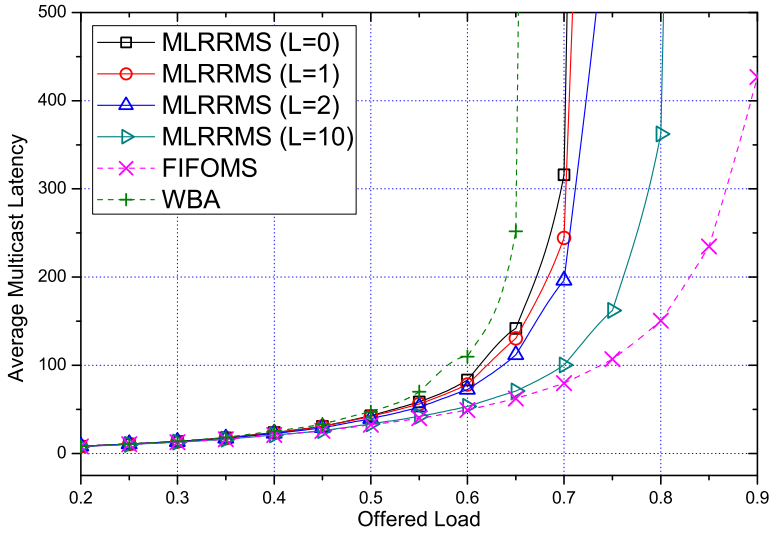


(a) $\omega = 0$, $N = 8$, $\bar{F} = 4$

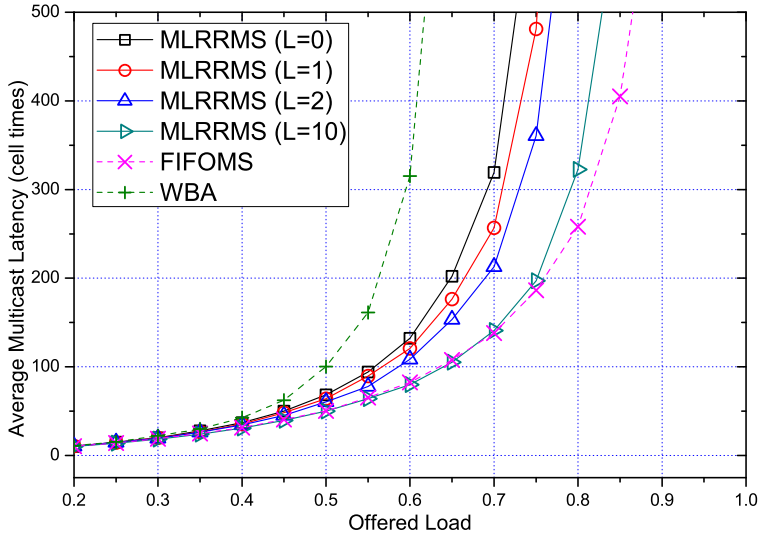


(b) $\omega = 0$, $N = 32$, $\bar{F} = 16$

Figure 4.16: Average look-ahead depth under bursty traffic (cell-based fan-out mode).

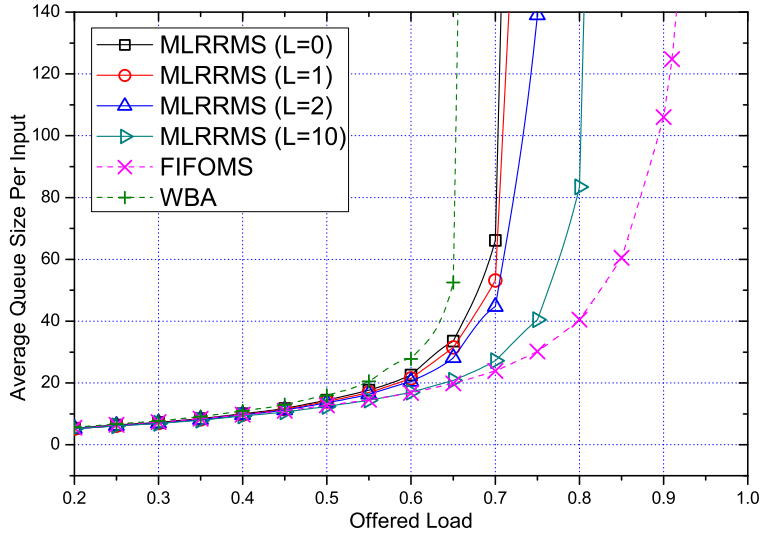


(a) $\omega = 0$, $N = 8$, $\bar{F} = 4$

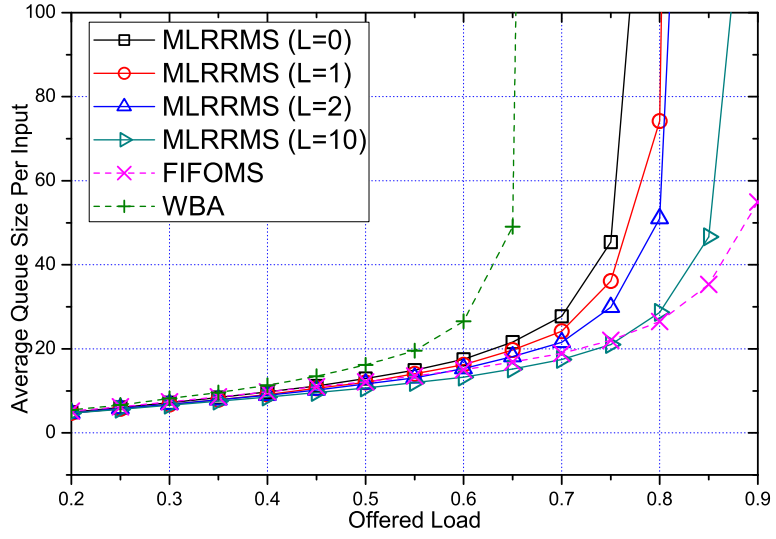


(b) $\omega = 0$, $N = 32$, $\bar{F} = 16$

Figure 4.17: Average multicast latency under bursty traffic (burst-based fan-out mode).

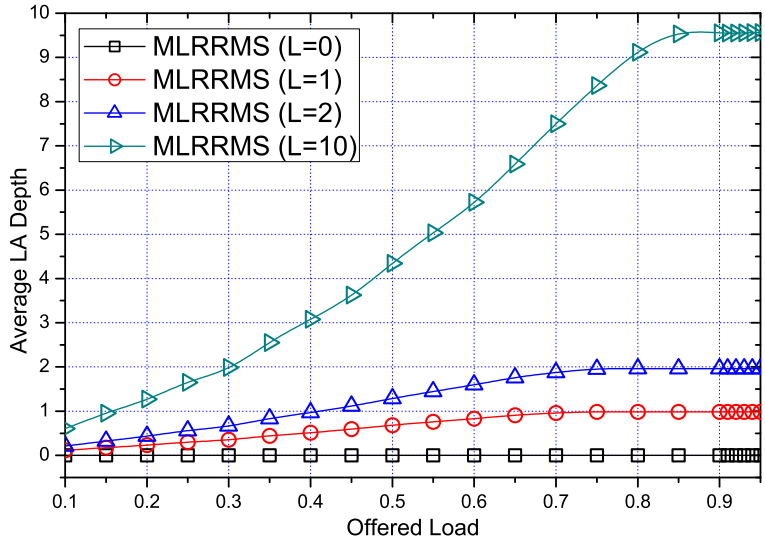


(a) $\omega = 0$, $N = 8$, $\bar{F} = 4$

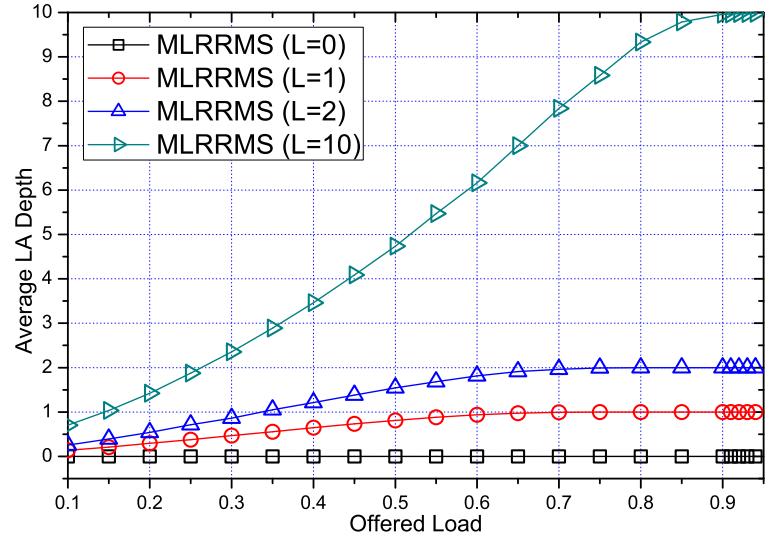


(b) $\omega = 0$, $N = 32$, $\bar{F} = 16$

Figure 4.18: Average queue size per input under bursty traffic (burst-based fan-out mode).



(a) $\omega = 0$, $N = 8$, $\bar{F} = 4$



(b) $\omega = 0$, $N = 32$, $\bar{F} = 16$

Figure 4.19: Average look-ahead depth under bursty traffic (burst-based fan-out mode).

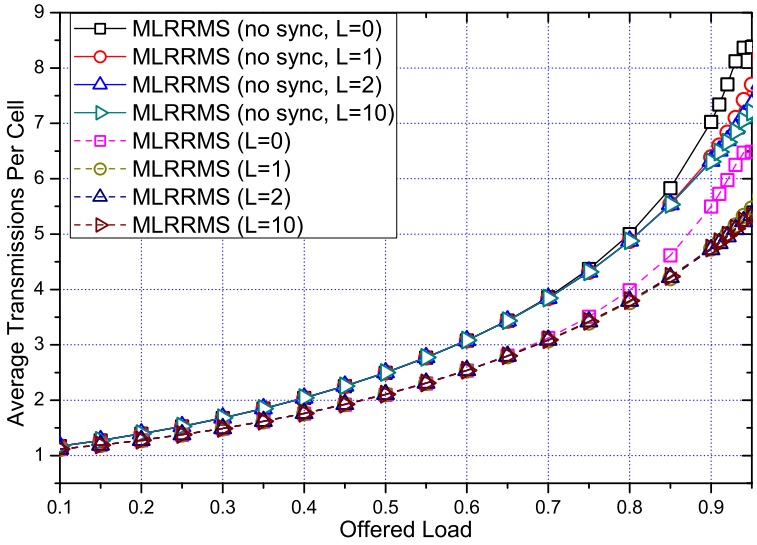


Figure 4.20: Improvement of the sync mechanism under Bernoulli traffic, with $\omega = 0$, $N = 32$, $\bar{F} = 16$.

In the MLRRMS algorithm, the sync mechanism aims to reduce the unnecessary multiple transmissions of cells. Figure 4.20 compares the average transmissions per cell under Bernoulli traffic for different L values, i.e. $L = 0, 1, 2, 10$. An obvious difference between the MLRRMS schemes with sync and without sync mechanism can be observed in the results. At the offered load of 0.8, for instance, the sync mechanism can reduce the average transmissions per cell by 1, which results in a significant reduction of the cells traversing the switch fabric.

Figure 4.21 compares the average transmissions per cell under bursty traffic with cell-based fan-out mode and Figure 4.22 compares the average transmissions per cell under bursty traffic with burst-based fan-out mode for different L values. The clear difference between the MLRRMS schemes with sync and without sync in Figure 4.21 is not obvious in Figure 4.22. This is due to the traffic profile of the burst-based fan-out mode where cells in the same burst carry the same fan-out vectors. Under the burst-based fan-out scheme, scheduling system needs to increase the L value in order to examine more cells. With a larger L, cells from

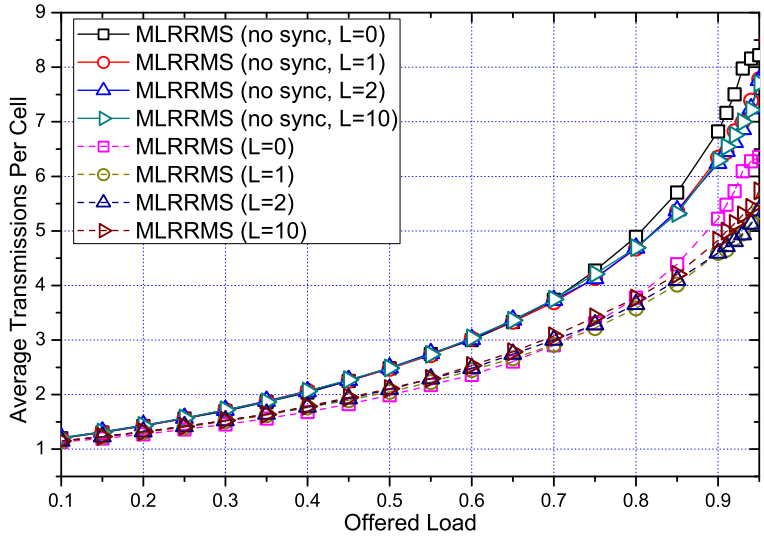


Figure 4.21: Improvement of the sync mechanism under bursty traffic (cell-based fan-out mode), with $\omega = 0$, $N = 32$, $\bar{F} = 16$.

a different burst can be transmitted, which results in a increase of the average transmissions per cell. This can be observed in between the MLRRMS (L=10) and the MLRRMS (no sync, L=0) in Figure 4.22.

4.6.3 Performance for Unbalanced Multicast Traffic under the Same Offered Load

When $\omega > 0$, the multicast traffic becomes unbalanced. Each bit of the fan-out vector has different probabilities of being 1 as in Formula 4.22. To examine how the balance factor ω affects the performance, we set the offered load fixed. From Formula 4.22, it can be calculated that when ω becomes larger than the threshold $\frac{N-E[F]}{E[F](N-1)}$, the actual mean fan-out will begin to decrease, due to the change of the Binomial distribution. The actual mean fan-out follows the formula shown in Formula 4.23:

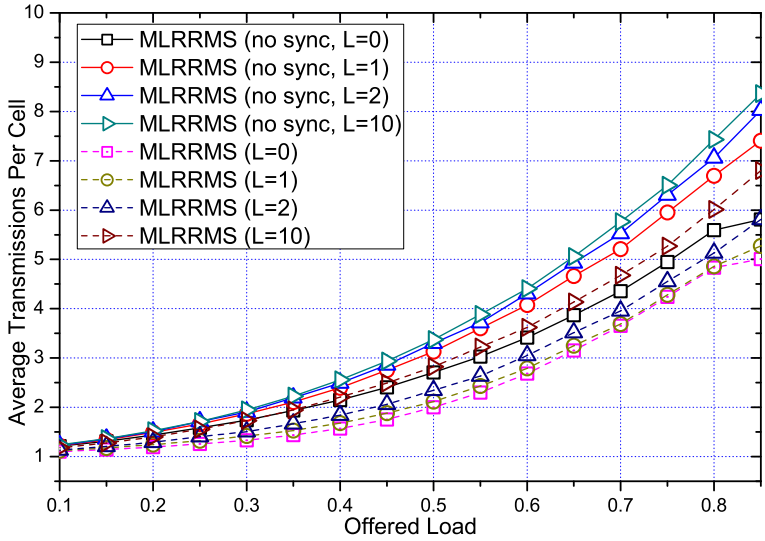


Figure 4.22: Improvement of the sycn mechanism under bursty traffic (burst-based fan-out mode), with $\omega = 0$, $N = 32$, $\bar{F} = 16$.

$$E[F]^* = \begin{cases} E[F], \left(0 \leq \omega \leq \frac{N-E[F]}{E[F](N-1)}\right) \\ \frac{E[F](N-1)(1-\omega)}{N} + 1, \left(\frac{N-E[F]}{E[F](N-1)} < \omega < 1\right) \end{cases} \tag{4.23}$$

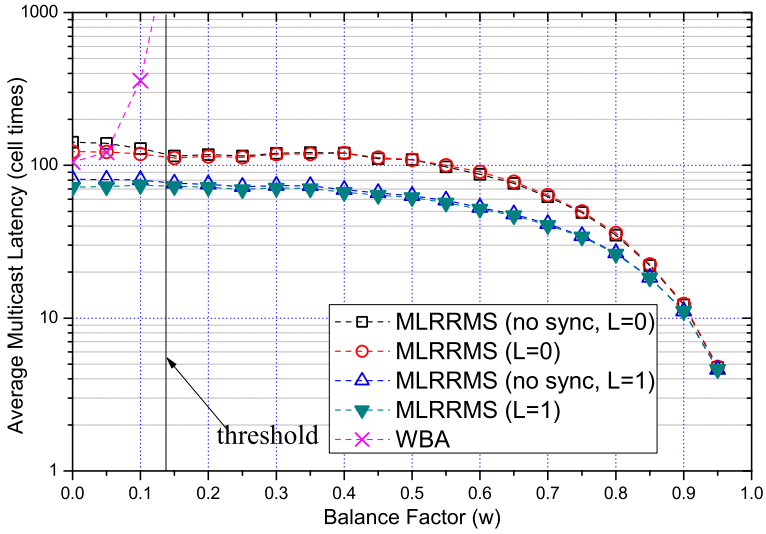
Thus, to achieve a fixed offered load, the arrival rate should be increased by a factor of $\frac{E[F]}{E[F]^*}$, in order to compensate the reduced mean fan-out. Unbalanced multicast traffic with $N = 8$ and $E[F] = 4$ is applied for performance evaluation.

Figure 4.23 shows the average multicast latency vs. the balance factor under a fixed offered load. The reason that he offered loads in Figure 4.23(a) and Figure 4.23(b) are different is that the maximum load that the switch can handle is reduced, when the burst-based fan-out mode is applied. In both Figure 4.23(a) and Figure 4.23(b), the multicast latency decreases with the increase of ω before it goes beyond the threshold in Formula 4.23. When ω becomes larger than the threshold, the arrival rate will increase to compensate the loss of $E[F]^*$. The

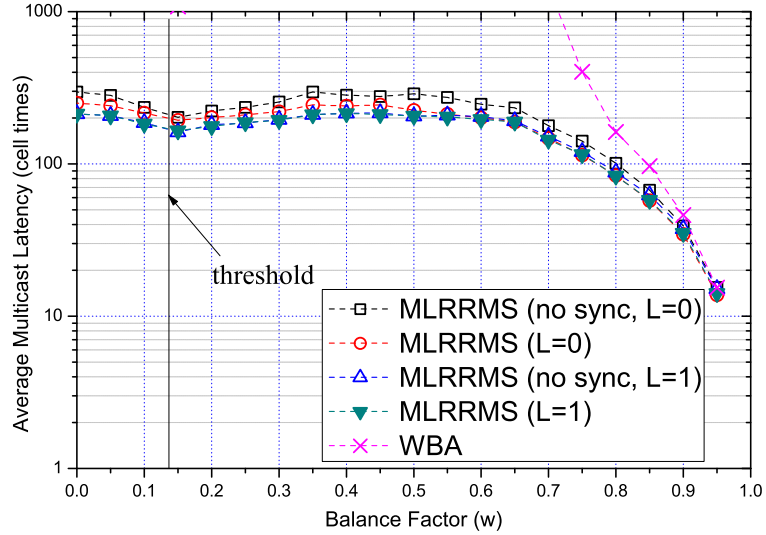
multicast latency initially increases, and then decreases as ω becomes larger. The system of WBA suffers from throughput degradation under unbalanced fan-out schemes, which will be shown later, and thus the focus is set on the comparison between the MLRRMS schemes. The results show that the MLRRMS (L=1) schemes have lower latency in comparison to the MLRRMS (L=0) schemes under various fan-out distributions in both fan-out modes. Before the threshold of ω, the delay curves drop as ω increases. This is due to the fact that traffic sent to input i begins to gather to output i, rather than spreading uniformly among all outputs. Traffic load on other outputs is therefore reduced, resulting in the decrease of latency. When ω reaches beyond the threshold, the arrival rate begins to increase and thus a rise of the curves can be observed in Figure 4.23. As ω continues to increase, a decrease of multicast latency occurs because the effect of the balance factor is more noticeable than the increasing arrival rate.

Figure 4.24 illustrates the average number of transmissions per cell vs. the balance factor under a fixed offered load. As ω increases, the number of transmissions decreases. When $\omega = 1.0$, the traffic becomes unicast traffic, and thus the number of transmissions per cell naturally becomes 1. As discussed previously, $E[F]^*$ decreases when ω crosses the threshold. This corresponds to the results in Figure 4.23. Jointly observing Figure 4.23 and Figure 4.24, we can discover that the LA mechanism causes an increase in the number of transmissions per cell in comparison to the basic schemes. This results in the decrease in the multicast latency. This is due to the fact that by searching into the queues, cells are more likely to receive service, rather than being blocked, which naturally increases the number of transmissions per cell. The HOL blocking problem is therefore alleviated, consequently reducing the average multicast latency.

Figure 4.25 compares the throughput under different balance factors with a fixed offered load. The MLRRMS schemes are able to maintain a stable throughput for various fan-out distributions. The throughput of the WBA is most affected by the change of the ω. It is important to maintain the throughput, for both balanced and unbalanced multicast traffic within the load that the switch is able to process. Throughput should be independent of the fan-out pattern, which is difficult to expect in reality. Otherwise the algorithm may suffer from the risk of sudden

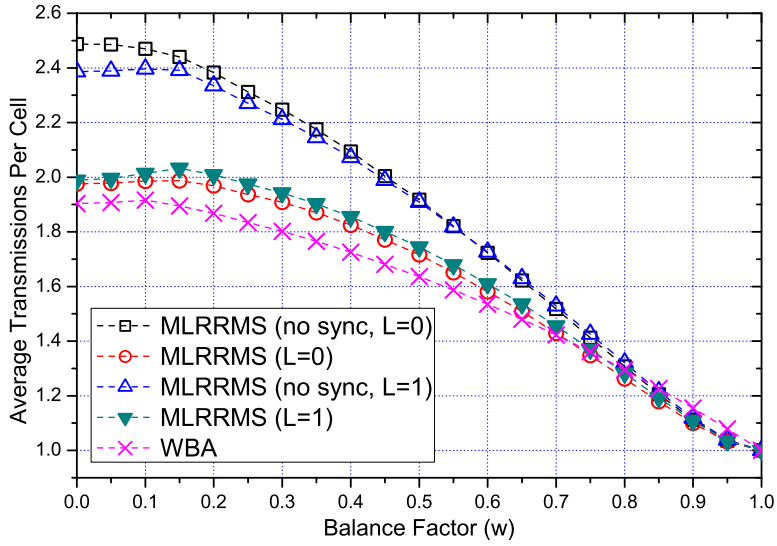


(a) Cell-based fan-out with output load of 0.8

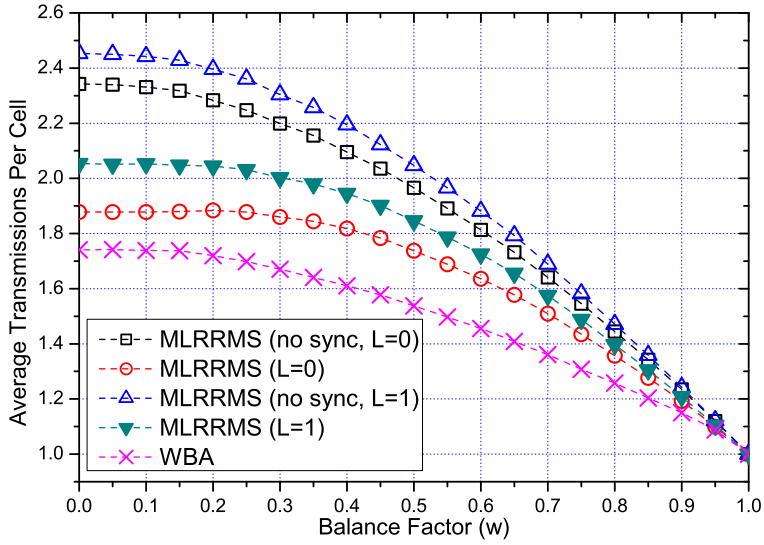


(b) Burst-based fan-out with output load of 0.68

Figure 4.23: Average multicast latency under different balance factors.



(a) Cell-based fan-out with output load of 0.8



(b) Burst-based fan-out with output load of 0.68

Figure 4.24: Average number of transmissions per cell under different balance factors.

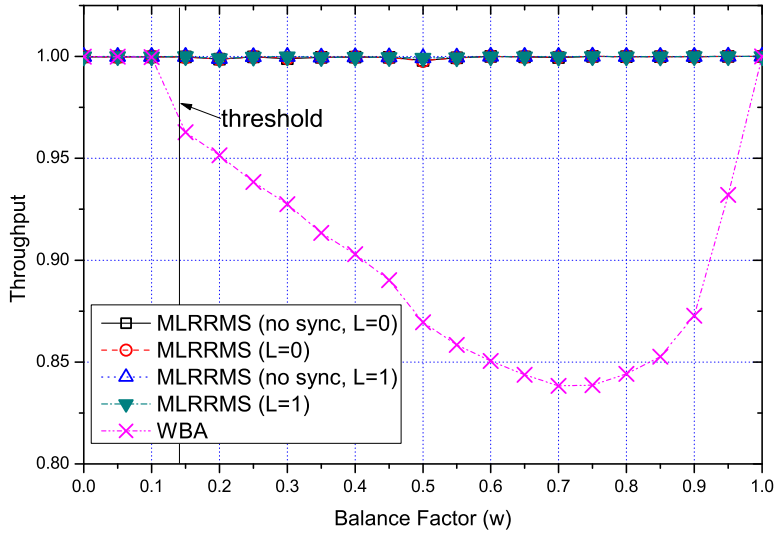
performance degradation. The proposed MLRRMS is able to meet such requirements as shown in Figure 4.25.

4.7 Summary

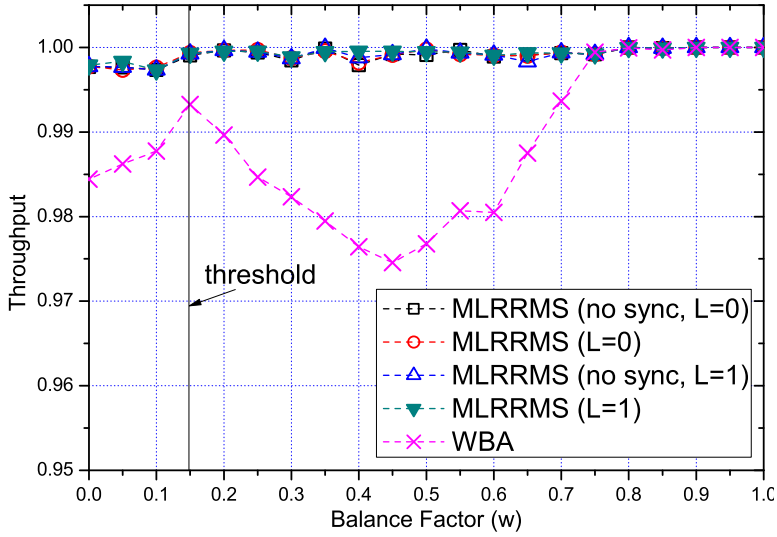
In this chapter, the MLRRMS algorithm with the LA mechanism and the sync mechanism is proposed for $N \times N$ switches of the FIFO-IQ architecture.

The LA and the sync mechanism, consisting of matrix operations used in MLRRMS, can be implemented in a parallel fashion with a low time complexity. Under varying traffic conditions, the MLRRMS with $L = 1$ outperforms the WBA and gains the largest performance improvement, compared to the added implementation complexity. The HOL blocking problem is alleviated by the LA mechanism. With a larger L, the algorithm performs close to the FIFOMS, which uses the VOQ structure, but the obtained marginal performance improvement does not justify the introduced implementation complexity.

In addition, under both balanced and unbalanced multicast traffic conditions, the MLRRMS with $L = 1$ is able to maintain a stable throughput compared to the WBA and achieves better performance than the MLRRMS with $L = 0$. Being able to search up to 1 cell stored further in the queues for the switch, i.e. being capable of processing 2 cells at the head of the queues, within one cell time, instead of creating multiple queues for each input, provides a significant performance improvement, in terms of multicast delay and average queue size in the practical switch design.



(a) Cell-based fan-out with output load of 0.8



(b) Burst-based fan-out with output load of 0.68

Figure 4.25: Throughput under different balance factors.

Chapter 5

Out-of-Sequence Prevention for Multicast Input-Queuing Space-Memory-Memory Clos-Network

To increase the scalability, multistage interconnection network is introduced to the switch fabric to reduce the number of crosspoints instead of the single-stage such as crossbar switch fabric. Since introduced by Charles Clos in 1953, the three-stage Clos architecture has constantly played an relevant role in the construction of high-scalability and large-capacity switches.

For the bufferless three-stage the bufferless central stage Clos-networks, often referred to as the Space-Space-Space (S^3) and Memory-Space-Memory (MSM) architecture, complex scheduling algorithms are required to avoid potential contentions due to the bufferless central stage. To reduce the scheduling complexity, buffers can be applied to the central stages. With a cell dispatching algorithm, the buffers at the input stages can be removed to reduce the implementation complexity. With buffers at the output stages, contentions are absorbed. This results in the Space-Memory-Memory (SMM) Clos-network architecture. How-

ever, cells traversing the fabric through different routes may experience different delays due to the introduced buffers, resulting in a disordered cell sequence when arriving at the outputs. This problem is referred to as the Out-Of-Sequence (OOS) problem. To reorder the disordered cells at the outputs, additional reassembly delays can be experienced and more buffers are required at the output ports for the processing. Multicast traffic is sensitive to this OOS problem because one single delayed traffic in the multicast group may ruin the application, such as Internet Protocol Television (IPTV) or teleconferencing.

In this chapter, an Input Queuing (IQ)-SMM architecture is proposed, aiming for high scalability. In order to prevent the OOS, two novel cell dispatching algorithms are proposed, i.e. the Multicast Flow-based DSRR (MF-DSRR) and the Multicast Flow-based Round-Robin (MFRR). Cells are no longer scheduled independently but treated as flows in order to maintain the order by both cell dispatching schemes. In comparison with the Desynchronized Static Round Robin (DSRR) [77], which treats cells independently, analytical and simulation results show that the MF-DSRR and MFRR outperform the DSRR in terms of reassembly delay and buffer size.

5.1 Introduction

Multicast handles traffic in a resource-efficient manner especially when it comes to bandwidth-intensive services such as IPTV or teleconferencing. A multicast router/switch can be implemented using a single crossbar switching fabric and various publications have discussed the scheduling algorithms for such a fabric type [7,60–62,75,78]. However the scalability of crossbar switches is limited by the growth of the number of cross points N^2, where N denotes the number of inputs/output ports, as shown in Figure 5.1. A Clos-network consists of several Switching Elements (SEs), as shown in Figure 5.2, and can be denoted as $C(n, m, r)$, where n is the number of input/output ports on the an Input Module (IM)/Output Module (OM), m is the number of Central Modules (CMs), and r is the number of IMs/OMs. Since each SE is a crossbar switch, the total number of cross points becomes $2nmr + mr^2$, where $nr = N$. By using multiple smaller crossbar switches, the Clos-network can reduce $N^2 - 2nmr - mr^2$ cross points when the number of input/output ports, N, is

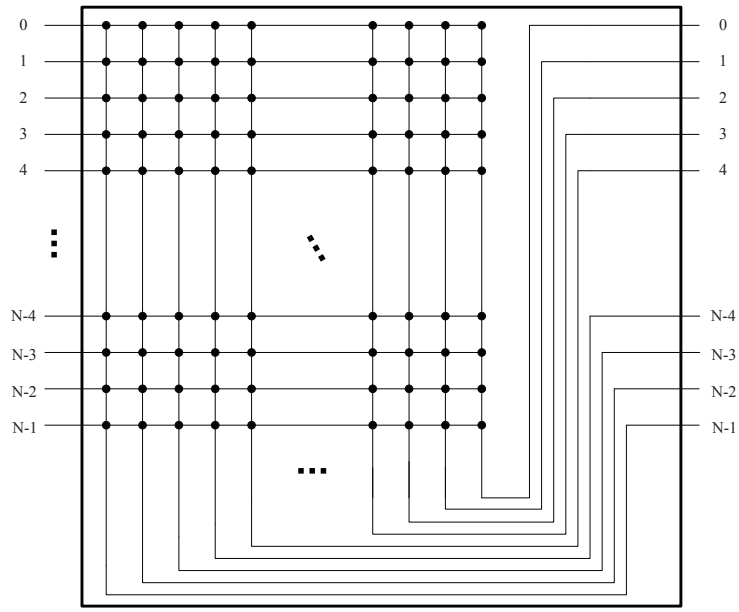


Figure 5.1: Crossbar switch fabric. The number of cross point is N^2, where N denotes the number of input/output ports.

large and is thus more scalable than crossbar switches [65, 79].

Usually, variable-length packets are segmented into several fixed-size cells before traversing the switch fabric. The IM is responsible for choosing one or several central switching modules to send the incoming multicast cells according to its Cell Dispatching (CD) algorithm. There are various CD algorithms, which are reviewed in Section 5.2, for different types of Clos-networks.

The MSM architecture, shown in Figure 5.3, refers to the architecture where memories are allocated at the output ports of IMs and the OMs. Since this architecture uses Output Queuing (OQ) for both IMs and OMs, it requires a speedup of the buffer, which hinders the scalability of the switch. A bufferless Clos-network, referred to as the S^3 architecture, has no speedup problem but a complex scheduling algorithm is required to solve the contention because the cells cannot be stored in any SE. It is obvious that a fully buffered architecture, referred to as the Memory-Memory-Memory (MMM) shown in Figure 5.4, has no

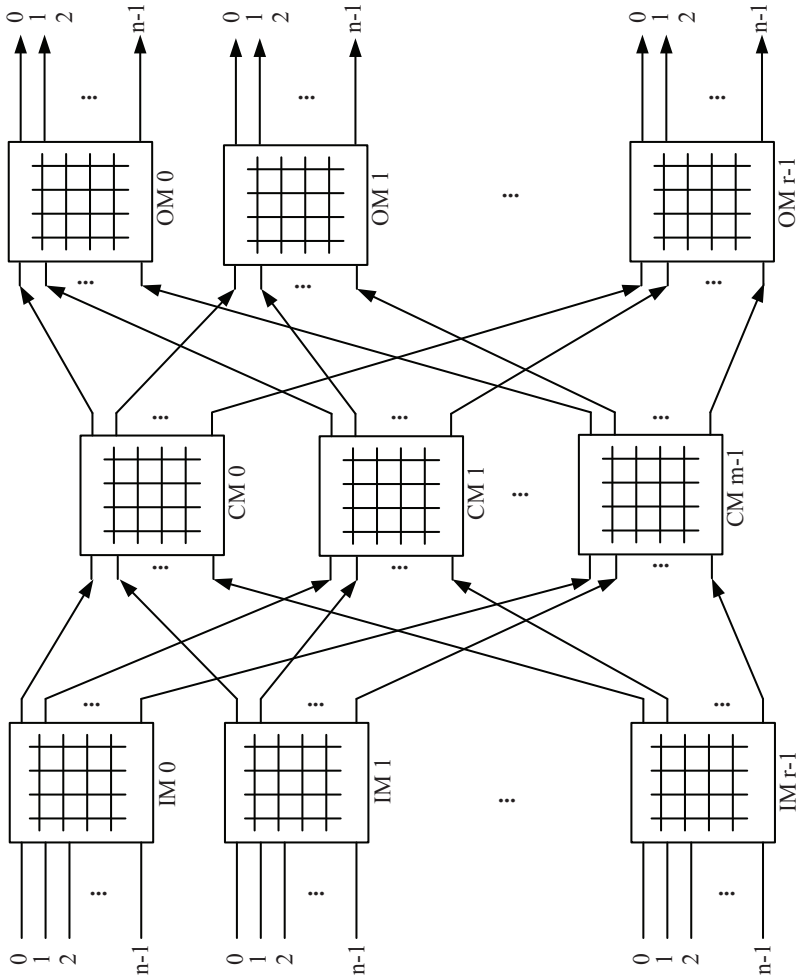


Figure 5.2: Clos-network switch fabric. The Clos-network consists of three stages of smaller-size switching elements.

contention problem and requires no complex scheduling algorithm but the implementation is costly since buffers are required for all the stages.

The SMM Clos-network architecture, proposed in [77], is shown to be able to achieve 100% throughput under admissible traffic. However, this architecture uses the OQ scheme at the CMs and the OMs, which limits the scalability due to the memory speedup problem. This architecture is referred to as the OQ-SMM in this chapter in order to be distinguished from the proposed IQ-SMM architecture. In addition to the speedup problem, as discussed, cells may experience different delays and arrives at the outputs out of order due to the buffers at the CMs if the CD algorithm is not carefully designed. Disordered cells need to be re-sequenced and reassembled to packets at the output ports. This requires extra memories and results in a longer delay. This problem is referred to as the Out-Of-Sequence (OOS) problem, which can be categorized into two types: *inter-packet* OOS and *in-packet* OOS. Inter-packet OOS means that cells generated by different multicast packets are disordered, and in-packet OOS implies that cells generated by the same multicast packet are disordered. The reason of such a subdivision is to differentiate the characteristics of OOS and further analyze and compare different CD algorithms so that appropriate reassembly methods can be applied further if required.

To the best of our knowledge, the OQ-SMM architecture proposed in [77] has not been evaluated for multicast. Therefore in this chapter, multicast traffic is be used to analyze the performance. In order to reduce the requirement of the internal speedup of the OQ-SMM and to solve the OOS problem, the IQ-SMM Clos-network with the Multicast Flow-based DSRR (MF-DSRR) and the Multicast Flow-based Round-Robin (MFRR) dispatching algorithms are proposed in this chapter. IQ crossbar switches, presented in Chapter 4, are leveraged as the CMs and OMs in the IQ-SMM architecture. The MF-DSRR/MFRR is independently run in the IMs, distributing incoming cells to the CMs. The MF-DSRR utilizes the connection pattern of the DSRR and takes multicast into consideration. The implementation complexity of the MF-DSRR is thus simple and it can alleviate the OOS problem. The MFRR, by using more resources, is able to eliminate the in-packet OOS problem and reduces the reassembly buffer size, and thus results in smaller reassembly delay compared to the DSRR and the MF-DSRR.

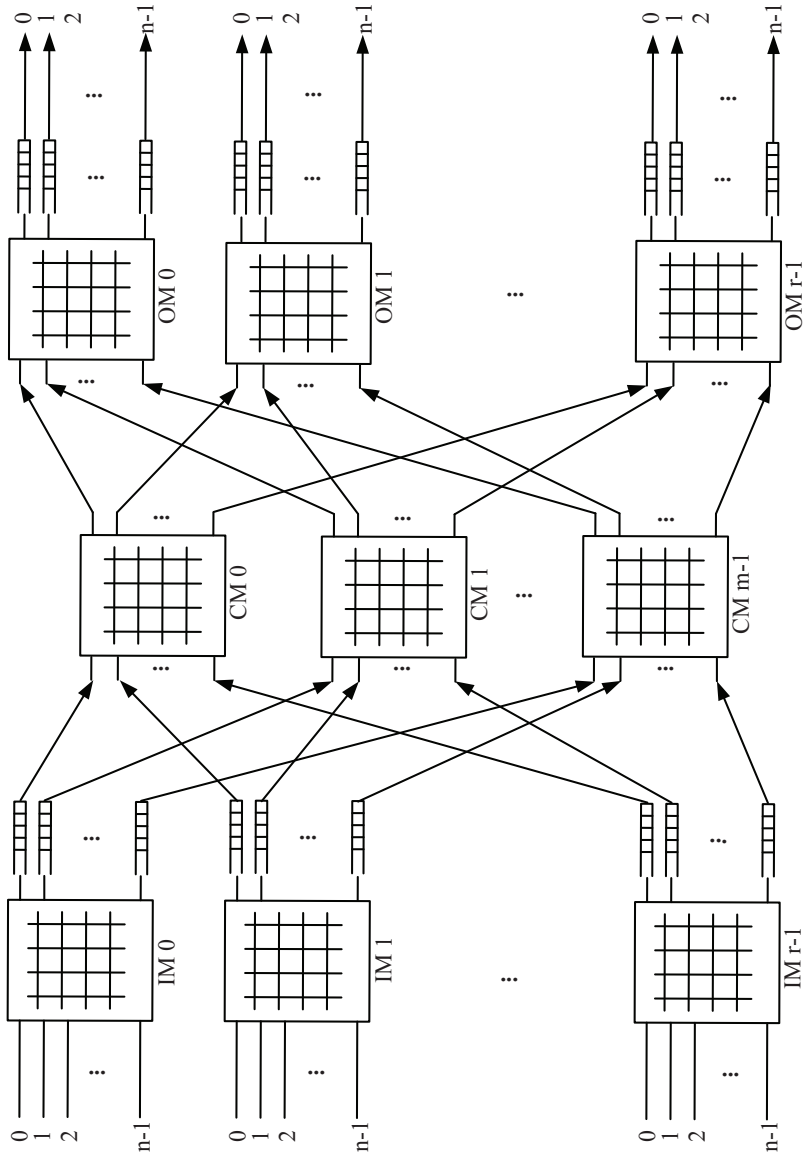


Figure 5.3: The Memory-Space-Memory (MSM) Clos-network architecture. Memories are placed at the outputs of the IMs and the OMs.

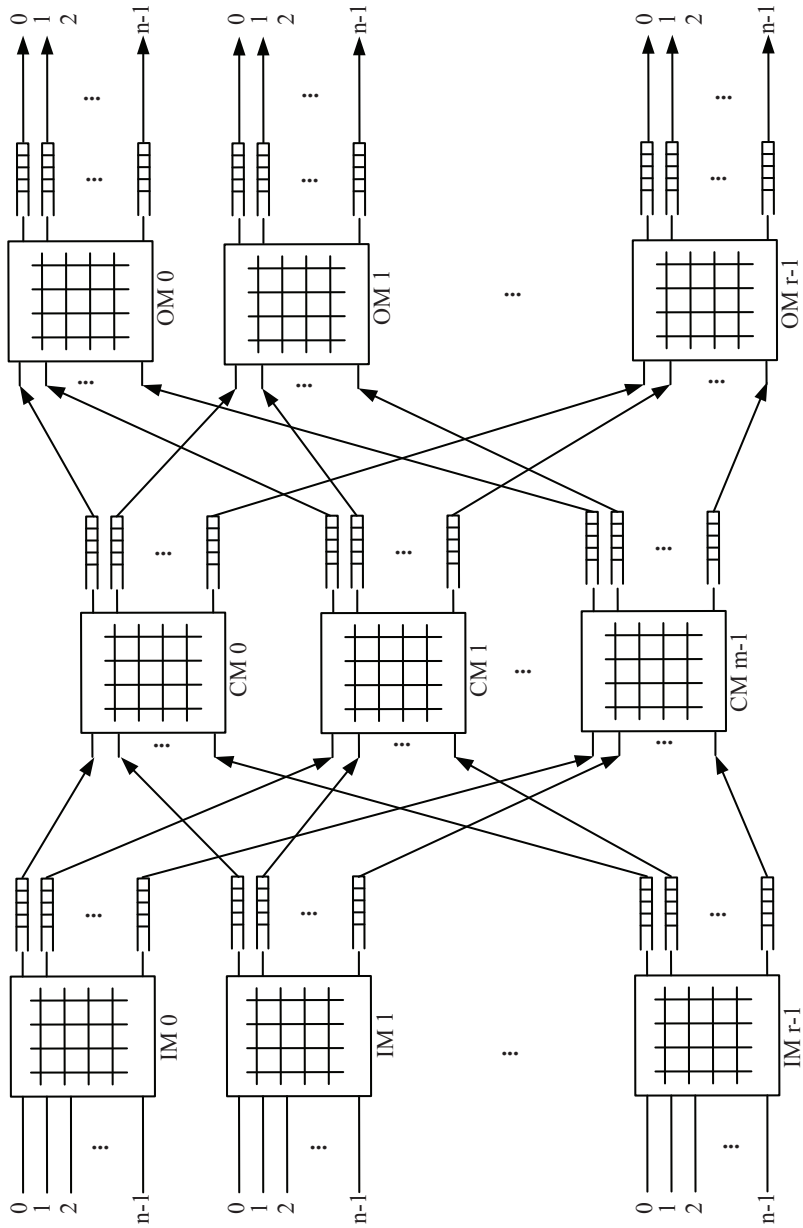


Figure 5.4: The Memory-Memory-Memory (MMM) Clos-network architecture. Memories are placed at the outputs of all switching elements.

The remaining parts of this chapter are structured as follows. In Section 5.2, different CD algorithms for various Clos-network architecture are reviewed. Section 5.3 describes the Clos-network model that is used throughout this chapter for the proposed CD algorithms. Section 5.4 introduces the MF-DSRR and the MFRR with detailed descriptions. Section 5.5 presents the analysis and shows the simulated performance. Section 5.6 concludes the chapter.

5.2 Related Work

Various scheduling algorithms have been developed for different Clos-network architectures with respect to buffer allocations. Table 5.1 compares the four existing architectures.

	Complex algorithm	**Memory speedup**	**OOS**	**Examples**
S^3	yes	no	no	Distro [80]
MSM	yes	yes	no	MWMD [81] CRRD/CMSD [82]
MMM	no	yes	yes	
OQ-SMM	no	yes	yes	DSRR [77]

Table 5.1: A comparison of different Clos-network architectures.

The architecture where memories are only allocated in the input/output stages is referred to as the MSM architecture. The Maximum Weight Matching Dispatching (MWMD) [81] and the Concurrent Round-Robin Dispatching (CRRD)/Concurrent Master-Slave round-robin Dispatching (CMSD) [82] are proposed for the MSM using round-robin arbitration. One disadvantage of these schemes is that both the input and output stages use shared memory schemes, resulting in a requirement of a speedup to the memory, which hinders the scalability of the switch. To solve this problem, a bufferless S^3 Clos-network architecture is proposed in [80]. Based on the Static Round-Robin Dispatching (SRRD), Distro [80] is proposed for the S^3 architecture and is demonstrated to achieve 100% throughout under uniform traffic. Both the MSM and the S^3 architectures require complex algorithms to solve the cell contention problems, since no buffers are allocated at the central stages.

In [77], the OQ-SMM Clos-network architecture with the DSRR dispatching is proposed. The study demonstrates that the SMM can achieve 100% throughput with the DSRR under admissible traffic. However, the OQ-SMM with DSRR in [77] uses OQ for both buffered stages, where speedup is again required. Besides, since memories are placed in the central stages, cells can experience different delays and thus resequencing is required to solve the OOS problem at the output port. When concerning multicast, the DSRR, initially designed for unicast and not evaluated for multicast, can result in a serious OOS problem, causing different delays within the same multicast group. This strongly affects the performance of some multicast applications, e.g. IPTV, since users of the same group experience different delays.

5.3 System Model

Assume that a First-In-First-Out (FIFO) queue is installed at each input port to temporarily store multicast packets. Packets are assumed to be segmented into fixed-size cells in the Input Port Processors (IPPs) before entering into the IMs and to be reassembled at the Output Port Processors (OPPs) after traversing the switch fabric. The IQ-SMM Clos-network switch model consists of three stages of SEs and is denoted as $C(n, m, r)$. As shown in Figure 5.5, the switch has r IMs/OMs of size $n \times m$, m CMs of size $r \times r$. Each IM/OM has n connections to IPPs, and m interstage connections to CMs. Only one interstage connection exists between an IM/OM and a CM. The number of input/output ports of the switch is $N = nr$. Cell dispatching algorithms are run by the IMs. Buffers are placed at the inputs of the CMs and the OMs to store the incoming cells. A CM or an OM runs the Multi-Level Round-Robin Multicast Scheduling (MLRRMS) algorithm proposed in Chapter 4. All CMs and OMs have the same maximum look-ahead depth. Each buffered SE can be considered as an IQ switch proposed in Chapter 4.

Since the IQ-SMM architecture is applied, the IMs are bufferless and forward incoming cells to the CMs following certain cell dispatching algorithms, which are investigated in the following paragraphs. CMs and OMs are IQ crossbar switches. Each CM has r FIFO input queues and each OM has m FIFO input queues, each of which is connected to

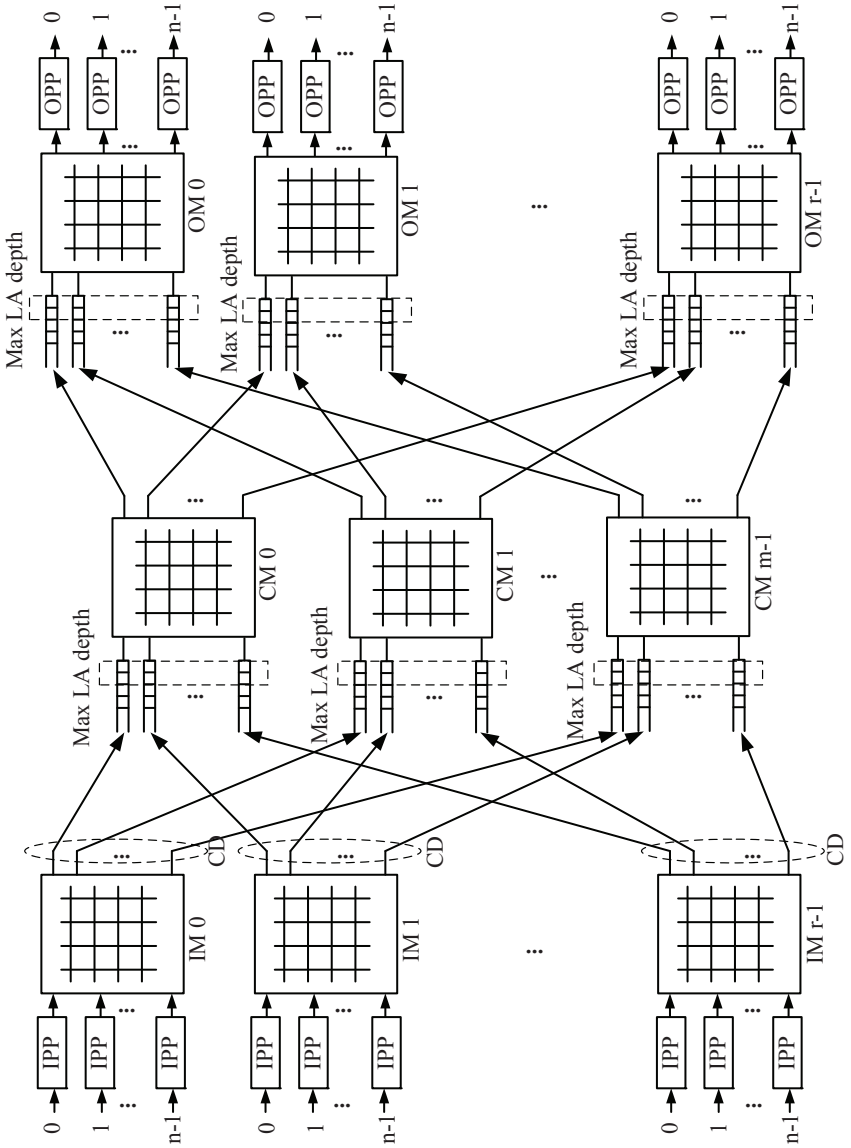


Figure 5.5: The Input-Queued Space-Memory-Memory (IQ-SMM) Clos-network architecture. Buffers are placed at the inputs of the central modules and the output modules.

one interstage link. Several notations used throughout this chapter are listed as follows:

IM_i: i^{th} input module, where $0 \leq i \leq r-1$

CM_k: k^{th} central module, where $0 \leq k \leq m-1$

OM_j: j^{th} output module, where $0 \leq j \leq r-1$

$I_{i,p}$: p^{th} input port of IM_i, where $0 \leq p \leq n-1$

$O_{j,q}$: q^{th} output port of OM_j, where $0 \leq q \leq n-1$

$Q_{C_{i,k}}$: i^{th} input queue of CM_k connected to IM_i

$Q_{O_{k,j}}$: k^{th} input queue of OM_j connected to CM_k

$IL_{i,k}$: the interstage link connecting IM_i and CM_k

$CL_{k,j}$: the interstage link connecting CM_k and OM_j

Assume that each packet carries a fan-out vector $\mathbf{b} = \langle b_j \rangle$, $b_j \in \{0,1\}$, $0 \leq j \leq N-1$, where $b_j = 1$ indicates that the packet is destined to the j^{th} output port, otherwise $b_j = 0$. An example of the fan-out vector is shown in Figure 5.6. Cells generated from the same packet have the same fan-out vectors. Cells are replicated as far downstream as possible in the switch model. More specifically, no cell replication occurs in the IMs, in contrast, multicast capability is implemented in the CMs and OMs. The FIFO queue is assumed to be able to examine the fan-out vector of each packet and inform the switch fabric of any fan-out change.

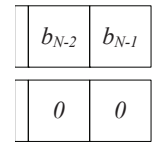


Figure 5.6: Demonstration of a fan-out vector of N bits. The cell carrying this the fan-out vector shown in this figure is bound for output port 1, 2, and N-3, since the bits on those positions are 1.

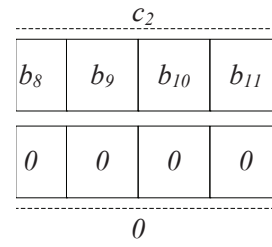


Figure 5.7: An example of the bit-cluster. The fan-out vector has $N = 12$ bits, and each bit-cluster has $n = 4$ bits. Therefore the fan-out vector can also be expressed by 3 bit-clusters. The cell is sent to OM_0 and OM_1 accordingly.

A bit-cluster is defined as a set of n consecutive bits in the fan-out vector, and is denoted as $\mathbf{c}_d = \langle b_j \rangle$, $0 \leq d \leq r-1$, and $nd \leq j \leq nd+n-1$. A fan-out vector consists of r non-overlapping bit-clusters and therefore the intersection of different bit-clusters is empty, i.e. $c_{d_1} \cap c_{d_2} = \emptyset$, if $d_1 \neq d_2$. Thus, a fan-out vector can also be expressed by bit-clusters, $\mathbf{b} = \langle \mathbf{c}_d \rangle$, $0 \leq d \leq r-1$. Define that $|\mathbf{c}_d| = \min\left(1, \sum_{j=nd}^{nd+n-1} b_j\right)$. The CM_k transmits an incoming cell by examining r bit-clusters of the fan-out vector. If $|\mathbf{c}_d| \neq 0$, CM_k sends a copy of the cell to OM_d. The OM_d examines all the bits in $\mathbf{c}_{d-1}$, and if $b_j = 1$, the OM_d sends a copy of the cell to $O_{d,j-nd}$. Figure 5.7 shows an example of the bit-cluster with $n = 4$ in a fan-out vector with $N = 12$.

5.4 Cell Dispatching Algorithms

In general, a cell dispatching algorithm for an IM specifies which CM the incoming cells are sent to, following specific requirements such as balancing load or minimizing blocking probability. For the S^3 or the MSM Clos-network architectures, complex algorithm or strategies have been proposed in [83, 84] to avoid internal cell contentions. However in the SMM architecture, buffers are introduced in the CMs and OMs to resolve the internal cell contentions. Cells are temporarily stored in the buffers until the intended output links are available and thus no internal blocking is experienced. As discussed in [77], the DSRR runs independently in each IM and connects each input to all outputs in a

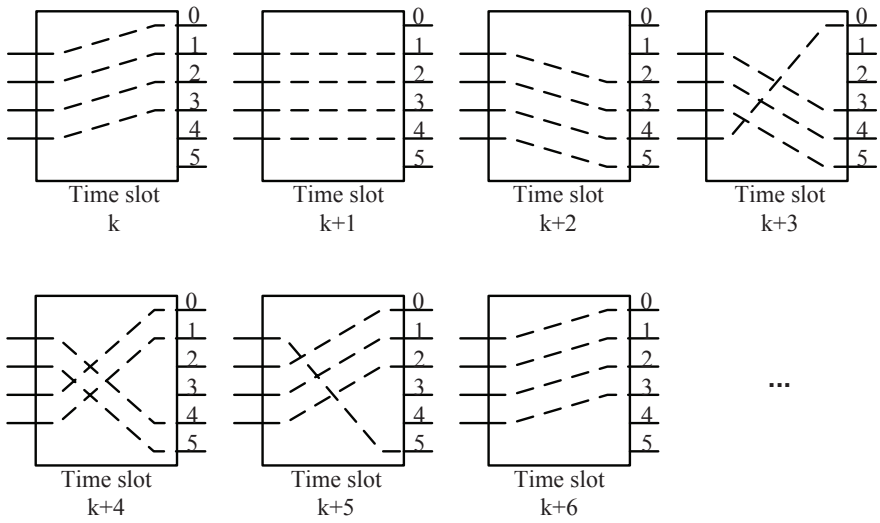


Figure 5.8: Desynchronized Static Round Robin connection pattern (DSRR). The connection pattern changes after each cell time.

round-round fashion. The connection pattern changes after each cell time as shown in Figure 5.8, resulting in a balanced distribution of cells to the CMs. However, when DSRR is applied to the IQ-SMM architecture shown in Figure 5.5, it causes a serious OOS problem since DSRR treats each cell independently. Therefore finding the cell dispatching algorithms that take the sequential order of incoming cells into account is of great relevance.

5.4.1 Multicast Flow-based Desynchronized Static Round-Robin (MF-DSRR) Dispatching

The MF-DSRR runs independently in each IM and requires that IM_i is notified when a change of received fan-out vector occurs. Unlike DSRR [77] that changes the IM connection pattern after every cell time, the MF-DSRR modifies the connections of all the input ports of the IM in a DSRR manner *when and only when* a notification is received. By utilizing the principle of the DSRR, the MF-DSRR maintains a low implementation complexity while reducing in-packet OOS problem. When

a new fan-out vector is detected, regardless on which input port the change occurs, each input port of the IM_i moves its connection to the interstage link next to the current one of the IM_i in a round-robin manner before a time slot begins. At most one change of the connection pattern is performed before a cell time.

The example shown in Figure 5.9 demonstrates how the connection pattern changes for a 4×6 IM. The initial configuration can be ($I_{i,0} \longrightarrow IL_{i,0}$, $I_{i,1} \longrightarrow IL_{i,1}$, $I_{i,2} \longrightarrow IL_{i,2}$, $I_{i,3} \longrightarrow IL_{i,3}$) as shown in Figure 5.9(1), where each input port is serving a flow of cells of the same fan-out vector. As a fan-out change occurs on $I_{i,0}$ from f_0 to $f_0^{'}$, each input port maps its connection to the next interstage link, resulting in a connection as ($I_{i,0} \longrightarrow IL_{i,1}$, $I_{i,1} \longrightarrow IL_{i,2}$, $I_{i,2} \longrightarrow IL_{i,3}$, $I_{i,3} \longrightarrow IL_{i,4}$), as shown in Figure 5.9(2). This connection pattern may be kept for several cell times until another change is detected in Figure 5.9(3), where $I_{i,3}$ has a fan-out change from f_3 to $f_3^{'}$. Then the connection pattern becomes ($I_{i,0} \longrightarrow IL_{i,2}$, $I_{i,1} \longrightarrow IL_{i,3}$, $I_{i,2} \longrightarrow IL_{i,4}$, $I_{i,3} \longrightarrow IL_{i,5}$). If another fan-out change occurs on $I_{i,2}$, then ($I_{i,0} \longrightarrow IL_{i,3}$, $I_{i,1} \longrightarrow IL_{i,4}$, $I_{i,2} \longrightarrow IL_{i,5}$, $I_{i,3} \longrightarrow IL_{i,0}$).

5.4.2 Multicast Flow-based Round-Robin (MFRR) Dispatching

The MFRR is independently run in each IM and also requires that each input port monitors the change of received fan-out vectors. Even though the MF-DSRR has a low implementation complexity, it fails to eliminate the in-packet OOS. During a transmission of a flow of cells of the same fan-out vector, it is possible that the connection pattern is changed due to a fan-out change detected on other input ports. This will cause distributing the same-packet cells (cells belonging to a same packet) to different CMs. In the MFRR, the change of connection pattern takes place independently on each input port, and thus eliminates the in-packet OOS problem.

A list called *AvailableList* is created for each IM to record the idle interstage links as its elements. Elements can be only popped from the top of the list and inserted to the bottom. When an element is popped, all the elements in the *AvailableList* are moved one position up to the top. Since there are n inputs and m interstage links in the IM, the

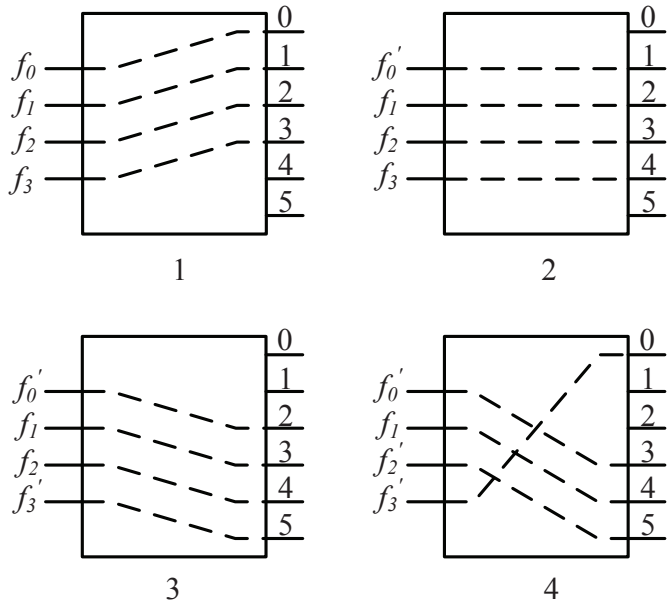


Figure 5.9: Multicast Flow-based DSRR cell dispatching (MF-DSRR). The connection pattern changes only when a fan-out change is detected. Each input port moves its connection to the interstage link next to the current one.

number of elements of the *AvailableList* is $(m - n)$. Therefore in order to use MFRR, the switch must have $m > n$.

Upon detecting a change of fan-out vector on $I_{i,p}$, IM_i pops an idle interstage link $IL_{i,k'}$ from the *AvailableList*, disjoins the connection $(I_{i,p} \longrightarrow IL_{i,k})$ and sets up a new connection $(I_{i,p} \longrightarrow IL_{i,k'})$ instead. The released interstage link $IL_{i,k}$ is inserted to the bottom of the *AvailableList*. For those input ports where no fan-out vector change is detected, the connections to the interstage links are unchanged. If multiple fan-out vector changes on several input ports in the same time slot are detected, ties are broken by randomly assigning the idle interstage links popped from the *AvailableList* to the input ports.

To better illustrate the scheme, a 4×6 IM shown in Figure 5.10 is considered. The connection pattern can initially be $(I_{i,0} \longrightarrow IL_{i,0}$, $I_{i,1} \longrightarrow IL_{i,1}$, $I_{i,2} \longrightarrow IL_{i,2}$, $I_{i,3} \longrightarrow IL_{i,3})$ with $\{IL_{i,4}, IL_{i,5}\}$ in the *AvailableList*. Assume a new fan-out vector $f_0^{'}$ occurs on $I_{i,0}$, $IL_{i,4}$ is popped from the list and a new configuration is established as $(I_{i,0} \longrightarrow IL_{i,4}$, $I_{i,1} \longrightarrow IL_{i,1}$, $I_{i,2} \longrightarrow IL_{i,2}$, $I_{i,3} \longrightarrow IL_{i,3})$. The released interstage link $IL_{i,0}$ is inserted to the bottom of the *AvailableList*, which becomes $\{IL_{i,5}, IL_{i,0}\}$. When a new fan-out vector is detected by $I_{i,3}$, $I_{i,5}$ is popped from the *AvailableList* and a connection pattern $(I_{i,0} \longrightarrow IL_{i,4}$, $I_{i,1} \longrightarrow IL_{i,1}$, $I_{i,2} \longrightarrow IL_{i,2}$, $I_{i,3} \longrightarrow IL_{i,5})$ is established with the *AvailableList* being $\{IL_{i,0}$, $IL_{i,3}\}$. Further connection pattern modifications as fan-out vector changes are also shown in the figure.

5.5 Performance Analysis and Simulation Results

The traffic to each input port $I_{i,p}$ is assumed to be an independent *Poisson Arrival Process* with arrival rate of $\lambda \leq 1$. Variable-length packets are segmented into L fixed-size cells in the IPPs, where L is a random variable uniformly distributed with mean of $E(L) = \bar{L}$.

Packets are independent and each packet is bound for an output port $O_{j,q}$ with a probability of p:

$$P\left(b_j = 1\right) = p \tag{5.1}$$

The fan-out of a fan-out vector is defined as $F \triangleq |\mathbf{b}| = \sum_{j=0}^{N-1} b_j$.

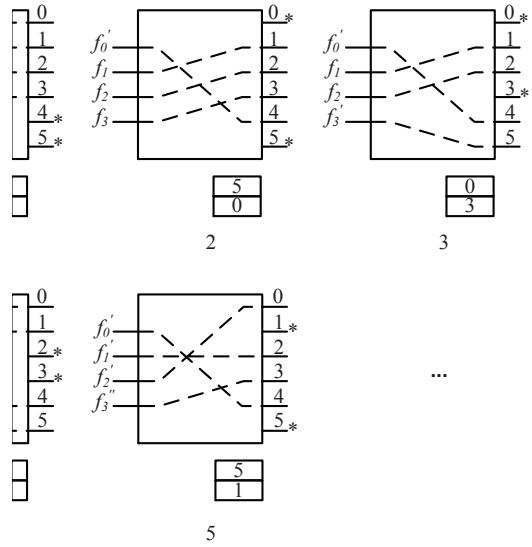


Figure 5.10: Multicast Flow-based Round Robin cell dispatching (MFRR). An *AvailableList* is maintained by each IM. When a fan-out vector change is detected, a link is popped from the top of the list and the input port moves its connection accordingly.

Since each packet is at least bound for one output port, the fan-out F for each packet is:

$$P(F=f)=\frac{\binom{N}{f}p^f(1-p)^{N-f}}{1-(1-p)^N} \tag{5.2}$$

$$E(F)=\bar{F}=\frac{Np}{1-(1-p)^N} \tag{5.3}$$

As described in Section 5.3, the CMs only observe the bit-clusters and the fan-out $F_{CM} \triangleq \sum_d |\mathbf{c}_d|, \forall d$, seen by a CM becomes:

$$P(F_{CM}=f)=\frac{\binom{r}{f}\left[1-(1-p)^n\right]^f\left[(1-p)^n\right]^{r-f}}{1-(1-p)^N} \tag{5.4}$$

$$E(F_{CM})=\bar{F_{CM}}=\frac{r\left[1-(1-p)^n\right]}{1-(1-p)^N} \tag{5.5}$$

All traffic is admissible, which implies that no input or output port is oversubscribed. The total packet traffic load on all output ports is $N\lambda\bar{F}$, and since traffic is equally distributed among all output ports, the offered packet load seen on each output is $\lambda\bar{F}$.

5.5.1 In-Packet OOS Performance of the MF-DSRR

The main principle of the MF-DSRR is to maintain the IM connection pattern until a change of fan-out vector occurs among the all traffic received by the IM. Intuitively, this low-complexity scheme cannot guarantee the elimination of in-packet OOS due to the varying packet length and unexpected packet arrivals. For an input port $I_{i,p}$ under the MF-DSRR cell dispatching, the probability of j connection pattern changes ($j=0,1,2,\ldots,\bar{L}$) during the transmission of a packet is:

$$P(j)=\binom{\bar{L}}{j}\left[(n-1)\lambda\right]^j\left[1-(n-1)\lambda\right]^{\bar{L}-j} \tag{5.6}$$

After m changes, the connection pattern of an IM resumes, thus the probability that same-packet cells are sent to different CMs is:

$$
\begin{aligned}
P^*_{MF-DSRR} &= 1 - \sum_{\theta} P(\theta) \\
&= 1 - \left(1 - \hat{\lambda}\right)^{\bar{L}} \cdot \\
&\quad \left[1 + \binom{\bar{L}}{m} \left(\frac{\hat{\lambda}}{1-\hat{\lambda}}\right)^{m} + \binom{\bar{L}}{2m} \left(\frac{\hat{\lambda}}{1-\hat{\lambda}}\right)^{2m} + \cdots \right]
\end{aligned}
\tag{5.7}
$$

where $\theta = 0, m, 2m, \ldots,$ and $\lambda = (n-1)\lambda$.

5.5.2 In-Packet OOS Performance of the MFRR

Unlike the DSRR and the MF-DSRR, the MFRR maintains the connection of each input port independently. Changes of fan-out vectors on an input port have no influence on others. Thus, during the transmission of same-packet cells, no connection interruption occurs. The probability that same-packet cells are sent to different CMs is:

$$P^*_{MFRR} = 0 \tag{5.8}$$

For the DSRR dispatching scheme, the connection pattern changes every cell time. Therefore the probability that same-packet cells are sent to different CMs is:

$$P^*_{DSRR} = P\left(L > 1\right) = 1 - P\left(L = 1\right) = 1 - \frac{1}{L_{max}} \tag{5.9}$$

where L_{max} is the maximum number of cells contained in a packet.

The relation of the probability of same-packet cells being sent to different CMs among the DSRR, the MF-DSRR, and the MFRR thus becomes:

$$P^*_{DSRR} > P^*_{MF-DSRR} > P^*_{MFRR} \tag{5.10}$$

5.5.3 Time Complexity of MF-DSRR and MFRR

In the DSRR, each input port moves its connection to the next interstage link after each cell time. No complex algorithm is involved to establish

the new connection pattern. Thus, the time complexity of a new connection pattern establishment in the DSRR is $O(1)$. Since the MF-DSRR leverages the DSRR, the time complexity of a new connection pattern establishment in the MF-DSRR is also $O(1)$.

The MFRR eliminates the in-packet OOS by using distributed and independent connection management for each input port and achieve low complexity introducing the *AvailableList*. Without using the *AvailableList*, each input port has to check the interstage link one by one until an idle one is found. In the worst case, an input port will look through $(m-1)$ interstage links, resulting in a time complexity of $O(m)$. With the *AvailableList*, an input port merely pops an element from the top of the list, and the time complexity of establishing a new connection of an input port is reduced to $O(1)$.

5.5.4 Advantages and Limitation of the MFRR

1) No Contention on Interstage Links

If each input port locally maintains a round-robin pointer without using the *AvailableList*, contentions on interstage links may occur. Multiple inputs choosing the same interstage link has to be solved. This can delay the establishment of the new connection and degrade the throughput of the IM. Since the IM is bufferless, cell loss can occur if throughput is degraded.

Using the *AvailableList*, the MFRR guarantees that each input port can always connect to an idle interstage link every time the input port detects a fan-out vector change. No computation with high complexity is required for the connection establishment and no cell loss occurs in the IM.

2) Fairness to CMs

If only local round-robin pointers are used for each input port, unfairness may occur. For an input port, the next available interstage link, e.g. $IL_{i,k}$ appears in the round-robin pointer can be the one which is released by another input port in the last cell time. In this case, the interstage link $IL_{i,k}$ will be consecutively busy for two multicast flows. In the worst case, $IL_{i,k}$ can be busy for n multicast flows, causing a sudden

cell increase in $Q_{C_{i,k}}$ and starvation in other queues.

Using the *AvailableList*, the MFRR can provide fairness among the interstage links. After releasing an interstage link, the IM waits $(m-n)$ times of fan-out vector changes before selecting the link again. This results in a fair distribution of different multicast flows to the CMs and no starvation occurs.

3) Memory Access Speed Requirement

The memory access speed of the *AvailableList* is required to be high enough to handle the n times of memory accesses (including read and write) within one cell time. If $n = 1$, the Clos-network needs N IMs, which is impractical from a scalability's perspective. When n becomes larger, the number of IMs reduces but the access speed of the *AvailableList* increases, which may lead to some implementational challenges if n becomes too large.

5.5.5 Simulation Results

The simulation is carried out in the OPNET Modeler [56]. The Static and the DSRR schemes are used as references in the performance comparison. The Static scheme is simply a stationary configuration of the internal connections of IMs, which keeps the same during the entire simulation.

Admissible traffic with $\bar{F} = 8$ and $\bar{L} = 13$ is generated independently at each input port of the simulated $C(4, 7, 4)$ IQ-SMM Clos-network switch. The multicast scheduling algorithms with sync to reduce multiple transmissions of cells used in the CMs and the OMs are described in [7,10], as well as in Chapter 4. A multicast cell may be served several times before it is removed from the queue. Unnecessary multiple transmissions can cause increased cell delays in a multicast switch. The sync mechanism aims to reduce the number of transmissions per multicast cells while maintaining the output port utilization. In order to reduce the Head-Of-Line (HOL) blocking problem, the IQ-SEs on both central and output stages can look ahead into the queues for cells that can be served to idle outputs. Since it is impractical to search an infinite depth into the queues, a maximum look-ahead depth, LA, is defined. If the

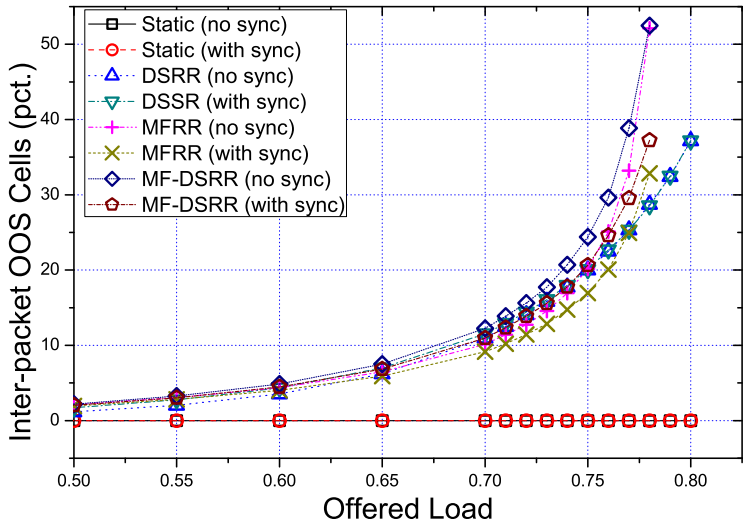


Figure 5.11: Percentage of inter-packet OOS cells, $LA = 0$.

LA is reached and the IQ-SE still has idle outputs, it stops the searching and completes the scheduling process.

Figure 5.11 compares the inter-packet OOS under different cell dispatching schemes with $LA = 0$. The counting of the inter-packet OOS cells is carried out in the OPP module shown in Figure 5.5. Under high offered load, more specifically when the load is beyond 0.77, the DSRR schemes (with and without sync) outperform the others except for the Static. This is because the DSRR evenly distribute cells to the input queues of the CMs and thus cells belonging to a packet are placed ahead of cells generated by another packet more than the MF-DSRR or the MFRR. It can also be observed that, with the sync mechanism, both MFRR and MF-DSRR can reduce the inter-packet OOS. This is due to that the sync mechanism aims to reduce the number of cell transmission without decreasing the output utilization of the switching module, which can result in the reduction of the inter-packet OOS cells.

Figure 5.12 compares the in-packet OOS under different cell dispatching schemes with $LA = 0$. The in-packet OOS cells are counted in the OPP module shown in Figure 5.5. Except Static, the MFRR

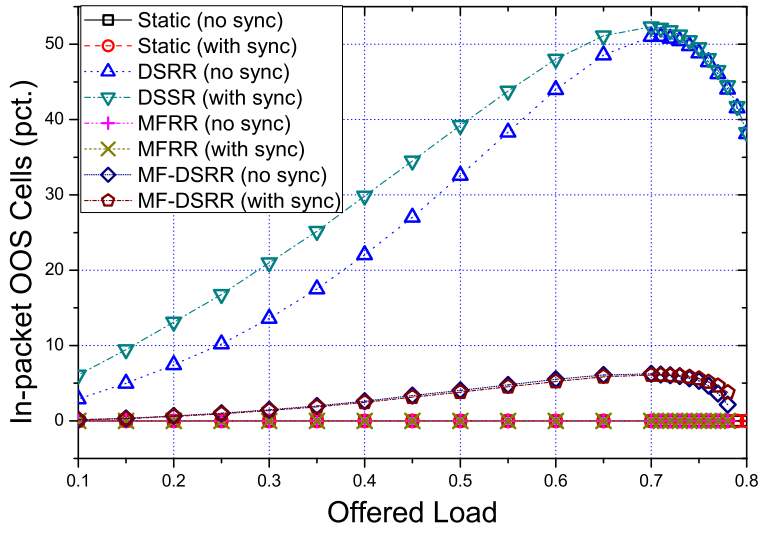


Figure 5.12: Percentage of in-packet OOS cells, $LA = 0$.

schemes outperform the others with zero in-packet OOS cells. Both the MF-DSRR schemes have less than 10% of total received cells being OOS cells under high offered load. The DSRR schemes result in serious in-packet OOS problems. With the sync mechanism, the DSRR causes more in-packet OOS cells. This is because, in the DSRR, same-packet cells are treated independently and are distributed to different CMs, resulting in a well distributed fan-out vectors in the input queues of the CMs and the OMs. Thus the same-packet cells have a higher probability to be disordered due to the round-robin working mechanism of the sync. A decrease of the in-packet OOS for the DSRR and the MF-DSRR schemes can be observed under the loads higher than 0.7. This is due to the fact that the CM queue length begins to increase non-linearly, resulting in more inter-packet OOS cells. Since the OPP modules consider inter-packet and in-packet OOS separately, the percentage of in-packet OOS therefore decreases.

Figure 5.13 shows the total number of OOS cells, i.e. the sum of inter-packet and in-packet OOS cells, under different schemes with $LA = 0$. The MFRR (with sync) significantly reduces the OOS problem under

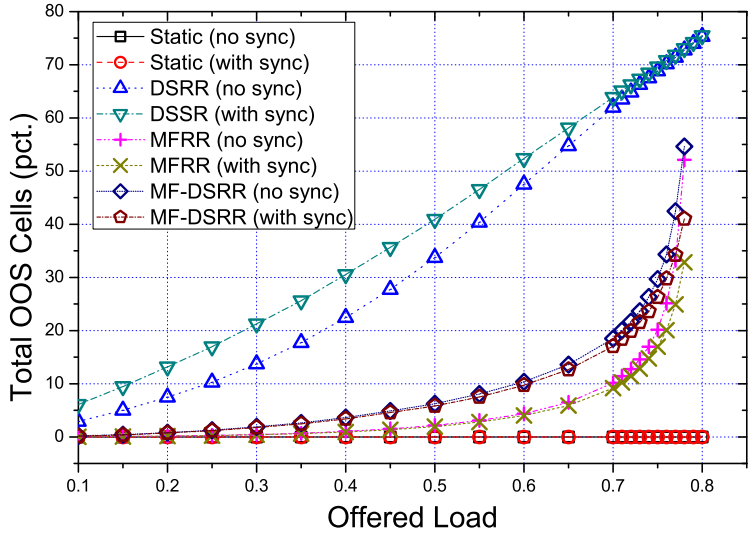


Figure 5.13: Percentage of the total number of OOS cells, $LA = 0$.

high offered loads, while the DSRR schemes cause nearly linear growths of the OOS cells with the increase of the offered load.

Figure 5.14 depicts the average reassembly delay ($LA = 0$) for each packet in cell times. The reassembly buffers are assumed to be located in the OPP module shown in Figure 5.5. Under heavy loads, the DSRR schemes result in 10 cell times of the reassembly delay, which is about 77% of mean packet transmission time. The MFRR (with sync) reduces the delay to approximately 3.5 cell times under heavy loads.

Figure 5.15 shows the average reassembly buffer size ($LA = 0$). The average buffer size of the Static schemes become stable under high offered load because the buffers at the OMs become unstable and the throughput reduces. The DSRR schemes require larger reassembly buffers at the OPPs under offered loads larger than 0.5. The MFRR schemes are able reduce the average reassembly buffer size.

Besides the average reassembly buffer size, the maximum buffer size is also worth examining, since it can be used as a benchmark in designing the reassembly buffer. Figure 5.16 compares the maximum reassembly buffer size. The DSRR and the MF-DSRR schemes demonstrate higher

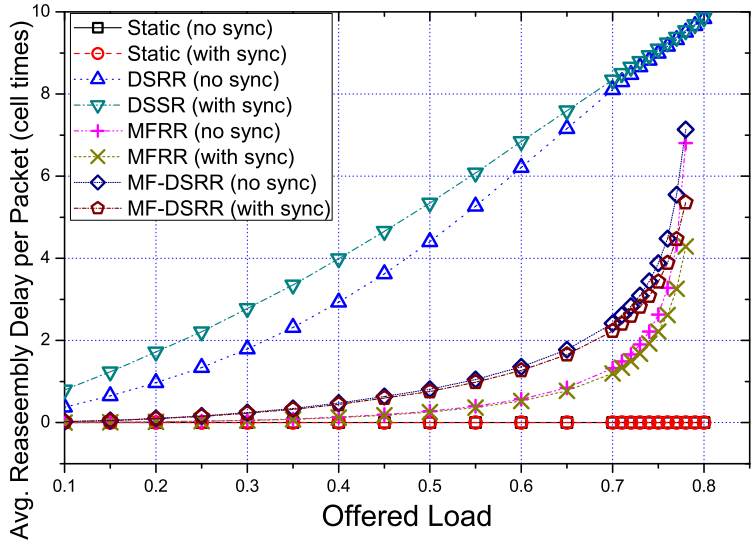


Figure 5.14: Average reassembly delay per packet, $LA = 0$.

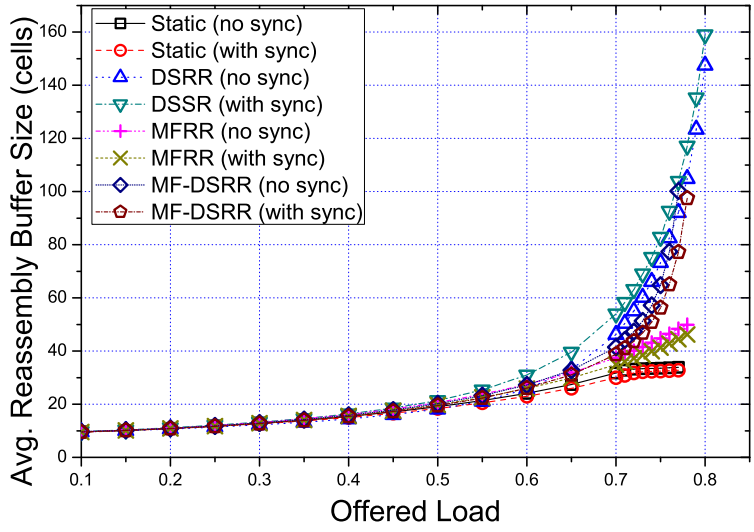


Figure 5.15: Average reassembly buffer size, $LA = 0$.

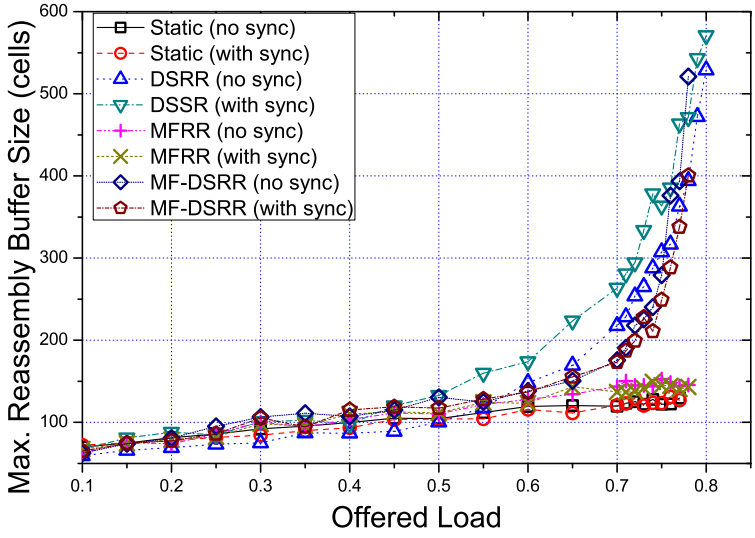


Figure 5.16: Maximum reassembly buffer size, $LA = 0$.

maximum buffer sizes, and the MFRR schemes are able to reduce the maximum buffer size.

Figure 5.17, Figure 5.18 and Figure 5.19 compare the inter-packet OOS, in-packet OOS and total OOS cells, respectively, under varying LA values with the sync mechanism. With larger LA values, a decrease on inter-packet OOS for each cell dispatching scheme is observed in Figure 5.17. The DSRR with $LA = 2$ outperforms the other two schemes. In Figure 5.18, the look-ahead mechanism greatly reduces the in-packet OOS for the DSRR scheme but the DSRR with $LA = 2$ still suffers from approximately 50% of all the received cells being in-packet OOS under high load. MFRR always maintains zero in-packet OOS under different LA values. In terms of the total OOS cells, the MFRR with $LA = 2$ outperforms the others, as shown in Figure 5.19. This is due to the fact that with the capability of looking ahead into the queues for blocked cells, the IQ-SEs are able to send some of those delayed cells that cause the OOS problem.

Figure 5.20 depicts the average assembly delay per packet under different LA values with the sync mechanism. The look-ahead mechanism

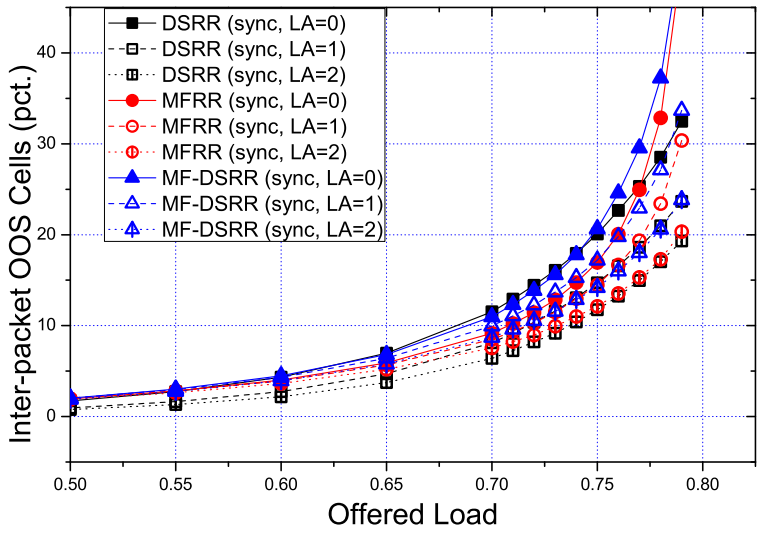


Figure 5.17: Percentage of inter-packet OOS cells, $LA = 0, 1, 2$.

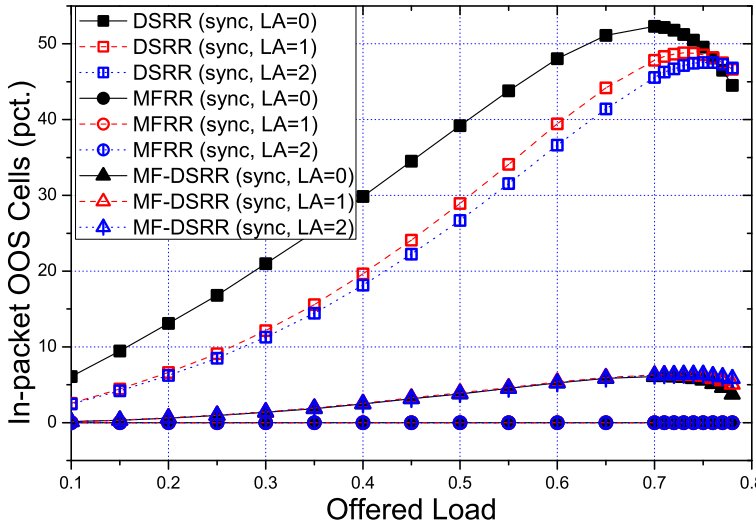


Figure 5.18: Percentage of in-packet OOS cells, $LA = 0, 1, 2$.

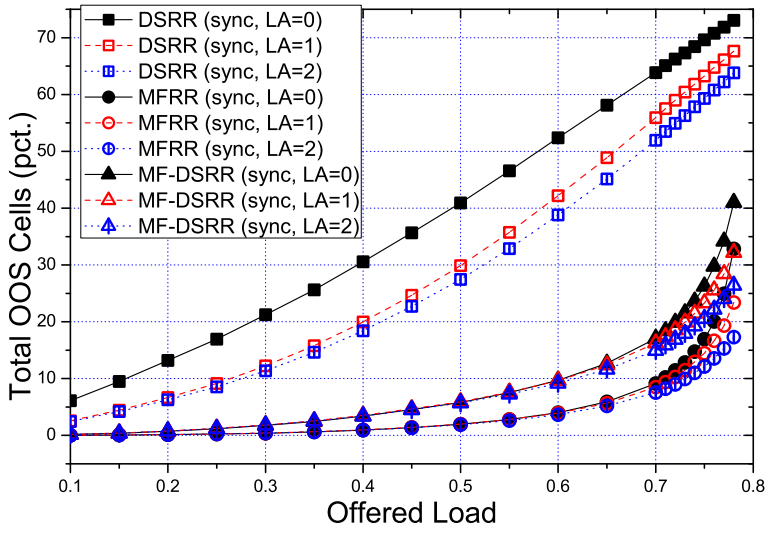


Figure 5.19: Percentage of the total number of OOS cells, $LA = 0, 1, 2$.

reduces the reassembly delay, which corresponds to the reduction of total OOS cells. The MFRR with $LA = 2$ outperforms the others.

Although the Static schemes have no OOS and low reassembly buffer sizes, they the highest cell delays, shown in Figure 5.21, because fewer CMs are used than other schemes. Static schemes become unstable after the offered load of 0.7, resulting in a decrease of the throughput, which explains the convergence in Figure 5.15. The DSRR schemes outperform the others in terms of average cell delay because cells are well distributed among the CMs. The MFRR schemes perform better than the MF-DSRR schemes under both sync options.

Figure 5.22 further compares the average cell delays of the DSRR, the MFRR and the MF-DSRR under different LA values with the sync mechanism. As discussed previously, the look-ahead mechanism applied in both CMs and OMs reduces the cell delay. The DSRR with $LA = 2$ has the lowest cell delay due to feature of evenly distributing cells to the CMs..

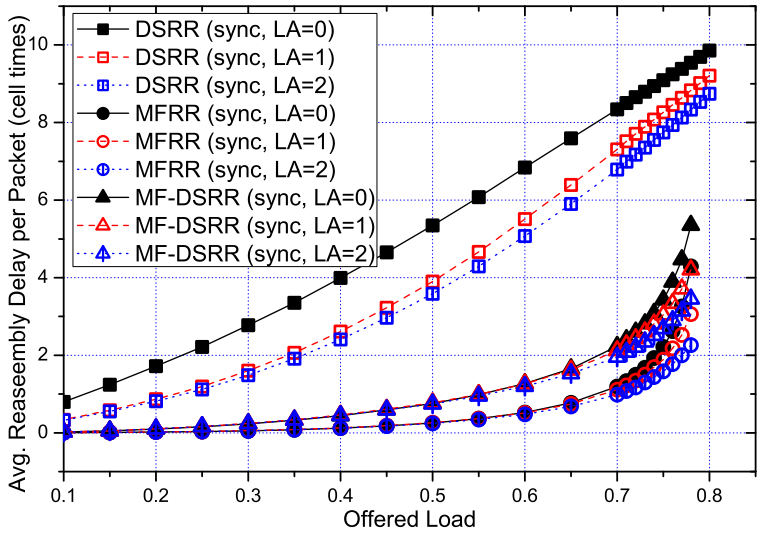


Figure 5.20: Average reassembly delay per packet, $LA = 0, 1, 2$.

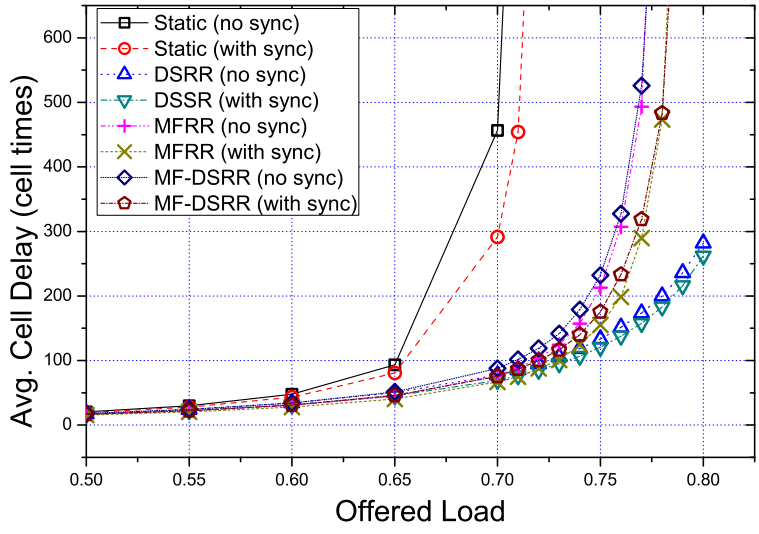


Figure 5.21: Average cell delay, $LA = 0$.

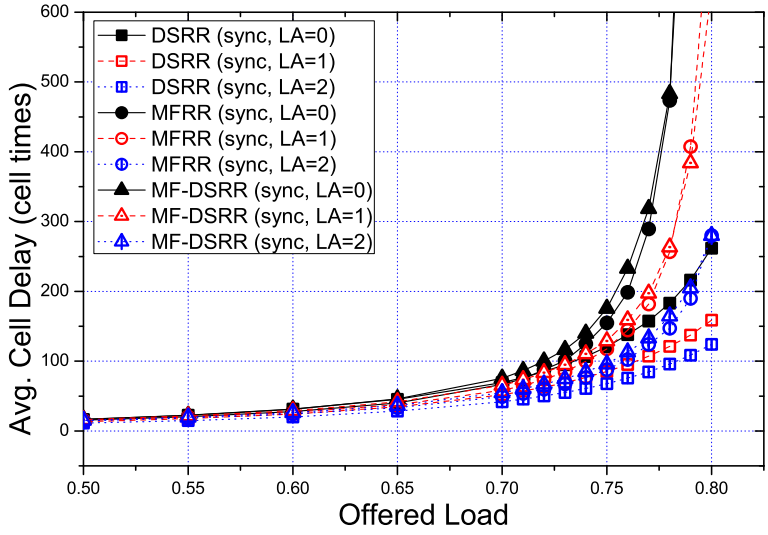


Figure 5.22: Average cell delay, $LA = 0, 1, 2$.

5.6 Summary

In this chapter, two OOS preventative cell dispatching algorithms are proposed for the multicast IQ-SMM Clos-network architecture, i.e. the Multicast Flow-based DSRR (MF-DSRR) and the Multicast Flow-based Round-Robin (MFRR). Table 5.2 presents a summarized comparison of different Clos-network architecture.

	Complex algorithm	**Memory speedup**	**OOS**	**Examples**
S^3	yes	no	no	Distro [80]
MSM	yes	yes	no	MWMD [81] CRRD/CMSD [82]
MMM	no	yes	yes	
OQ-SMM	no	yes	yes	DSRR [77]
IQ-SMM	no	no	yes	MF-DSRR, MFRR

Table 5.2: A summarized comparison of different Clos-network architectures.

The MF-DSRR utilizes the connection modification pattern of the DSRR and obtains a low implementation complexity. The MF-DSRR can alleviate the OOS problem but still suffers from the in-packet OOS problem. Using more resources, i.e. the *AvailableList*, the MFRR is able to eliminate the in-packet OOS problem and thus significantly reduces the reassembly buffer size and delay. With the use of the *AvailableList* to store the information of the idle interstage links, the MFRR can achieve a low complexity for internal connection setup of IMs.

Simulation results show that the MFRR cell dispatching outperforms the DSRR and the MF-DSRR in terms of reducing the OOS problem, the reassembly buffer size, and the reassembly delay. The sync mechanism is able to improve the performance of the MFRR and the MF-DSRR but worsens the performance of the DSRR in terms of the in-packet OOS. With the look-ahead mechanisms applied in both CMs and OMs, the IQ-SMM architecture can further reduce the OOS problem and the cell delay.

Chapter 6

Conclusion

Since the boom of high-bandwidth applications, such as Internet Protocol Television (IPTV), telecommunication service providers have gone through the continuous increase of bandwidth requirement. In both wireless and wired communication networks, the access speed experienced by customers has increased by a factor of more than 100 in the past decades. This trend has started the era of 100 Gigabit Ethernet in the next generation transport network.

In this dissertation, traffic management for the next generation transport network is investigated, in three different network scales. On the packet scheduling level, the topology-based hierarchical scheduling algorithm is proposed in Chapter 3. The proposed scheme is based on the assumption that, information of the network topology can be acquired by the scheduling system of the edge node. Token schedulers can be arranged by the scheduling system to map the acquired topology, in order to schedule the incoming traffic on behalf of the switches in the network, which lack of advanced traffic management abilities. This intelligent switch is usually place at the edge of the IPTV distribution network, so that the operator can leverage the already-built infrastructure to provide Quality-of-Service (QoS) guaranteed services. By network simulation, the topology-based hierarchical scheduling scheme demonstrates a strong flow isolation ability and an effective traffic management ability, comparing with the schemes that requires full network updates.

On the cell scheduling level, where the attention is mainly focused inside the switch, a novel Multi-Level Round-Robin Multicast Schedul-

ing (MLRRMS) algorithm is proposed for the input-queuing architecture in Chapter 4. Given the context of high capacity transport networks, the scalability and scheduling complexity become an extremely important issue. Thus, the Input Queuing (IQ) architecture is selected due to its high scalability. The proposed MLRRMS aims to surmount the Head-Of-Line (HOL) blocking problem of the IQ architecture and boosting the throughput of the switch. The sync mechanism is proposed to reduce the unnecessary multiple transmissions of a multicast cell, and the Look-Ahead (LA) process is used to reduce the HOL blocking problem. Analysis and simulation results show that with limited complexity the switch is proven able to achieve a high scalability and significant improvements in terms of multicast delay and throughput, compared to other existing multicast scheduling algorithms.

As the focus moves into the switch fabric, the three-stage Clos-network is investigated in Chapter 5. One of the challenges of multicast in the Clos-network is the prevention of Out-Of-Sequence (OOS) cells. Many literatures consider cells to be independent, however, it is not the case for most of the time. One packet usually generates more than one cells as it arrives at the switch fabric, where cell switching is mostly used to achieve high throughput. Therefore two OOS preventative cell dispatching schemes are proposed in Chapter 5 for IQ Space-Memory-Memory (SMM) Clos-network architecture, i.e. Multicast Flow-based DSRR (MF-DSRR) and Multicast Flow-based Round-Robin (MFRR). Analysis and simulation results demonstrate that, both the proposed schemes can reduce the OOS problem, resulting in a decrease of the reassembly delay and buffer size for the switch fabric.

The accomplishment achieved in this dissertation provides a guidance and a reference to the future research in traffic management for the next generation transport network. Firstly, the IPTV traffic management with topology-based hierarchical scheduling scheme can be further investigated. How to integrate the transport function with the control plane to make the scheduling system adapted to different network topologies and bandwidth allocation can be a research direction.

Secondly, multicast for switches is still open for discussion and research. Hardware implementation of the simulated scheduling algorithms will be an interesting topic, including performance evaluation, complexity analysis, and experiments. As the link speed increases to

100 Gbit/s, the packet processing time will become extremely short and therefore makes it challenging for hardware implementation of the advanced traffic scheduling system.

Last but not least, multicast inside the switch fabric, especially for the multi-stage switch fabric, needs further investigation. The cell dispatching schemes applied in the Clos-network should consider the route selection, in addition to the OOS prevention, by the means of back pressure or a control mechanism to ensure low cell delays and high throughput. Convergence of different switching technologies in the multistage switching network, such as time switching and space switching, is also an interesting area worth attentions.

Bibliography

[1] H. Yu, Y. Yan, and M. S. Berger, "IPTV traffic management in Carrier Ethernet transport networks," in *OPNETWORK 2008*, 2008.

[2] H. Yu, Y. Yan, and M. S. Berger, "IPTV traffic management using topology-based hierarchical scheduling in Carrier Ethernet transport networks," in *International Conference on Communications and Networking in China (ChinaCom)*, pp. 1–5, 2009.

[3] H. Yu, Y. Yan, and M. S. Berger, "Topology-based hierarchical scheduling using deficit round robin: Flow protection and isolation for triple play service," in *First International Conference on Future Information Networks*, pp. 269–274, 2009.

[4] A. Rasmussen, J. Zhang, H. Yu, R. Fu, S. Ruepp, H. Wessing, and M. S. Berger, "Towards 100 gigabit Carrier Ethernet transport networks," *WSEAS Transactions on Communications*, vol. 9, pp. 153–164, 2010.

[5] H. Wessing, M. S. Berger, H. Yu, A. Rasmussen, L. Brewka, and S. Ruepp, "Evaluation of network failure induced IPTV degradation in metro networks," *Recent Advances in Circuits, Systems, Signal and Telecommunications*, pp. 135–139, 2010.

[6] H. Wessing, M. S. Berger, H. M. Gestssson, H. Yu, A. Rasmussen, L. Brewka, and S. Ruepp, "Evaluation of restoration mechanisms for future services using Carrier Ethernet," *WSEAS Transactions on Communications*, vol. 9, pp. 322–331, 2010.

[7] H. Yu, S. Ruepp, and M. S. Berger, "A novel round-robin based multicast scheduling algorithm for 100 gigabit ethernet switches," in

29th IEEE International Conference on Computer Communications (INFOCOM) Workshops, pp. 1–2, 2010.

[8] H. Yu, S. Ruepp, and M. S. Berger, "Round-robin based multicast scheduling algorithm for input-queued high-speed Ethernet switches," in *OPNETWORK 2010*, 2010.

[9] H. Yu, S. Ruepp, and M. S. Berger, "Enhanced fifo based round-robin multicast scheduling algorithm for input-queued switches," *IET Communications*, vol. 5, pp. 1163–1171, 2011.

[10] H. Yu, S. Ruepp, and M. S. Berger, "Multi-level round-robin multicast scheduling with look-ahead mechanism," in *IEEE International Conference on Communications*, 2011.

[11] H. Yu, S. Ruepp, and M. S. Berger, "Out-of-sequence prevention for multicast input-queuing space-memory-memory Clos-network," *IEEE Communications Letters*, 2011.

[12] H. Yu, S. Ruepp, and M. S. Berger, "Out-of-sequence preventative cell dispatching for multicast input-queued space-memory-memory Clos-network," in *12th IEEE International Conference on High Performance Switching and Routing*, 2011.

[13] Y. Yan, H. Yu, and L. Dittmann, "Wireless channel condition aware scheduling algorithm for hybrid optical/wireless networks," in *3rd. International Conference on Access Networks*, pp. 397–409, 2008.

[14] Y. Yan, H. Yu, H. Wang, and L. Dittmann, "Integration of EPON and WiMAX networks: Uplink scheduler design," in *SPIE Symposium on Asia Pacific Optical Communications*, 2008.

[15] Y. Yan, H. Yu, H. Wessing, and L. Dittmann, "Integrated resource management for hybrid optical wireless (how) networks," in *International Conference on Communications and Networking in China (ChinaCom)*, pp. 1–5, 2009.

[16] Y. Yan, H. Yu, H. Wessing, and L. Dittmann, "Enhanced signaling scheme with admission control in the hybrid optical wireless (HOW) networks," in *28th IEEE International Conference on Computer Communications (INFOCOM) Workshops*, pp. 1–6, 2009.

[17] Y. Yan, H. Yu, H. Wessing, and L. Dittmann, "Integrated resource management framework in hybrid optical wireless networks," *IET Optoelectronics Special Issue on Next Generation Optical Access*, vol. 4, pp. 267–279, 2010.

[18] Metro Ethernet Forum, *http://metroethernetforum.org/*, 2011.

[19] L. Fang, R. Zhang, and M. Taylor, "The evolution of Carrier Ethernet services - requirements and deployment case studies," *IEEE Communications Magazine*, vol. 46, pp. 69–76, 2008.

[20] J. Mocerino, "Carrier class Ethernet service delivery migrating SONET to IP & triple play offerings," in *2006 Optical Fiber Communication Conference and National Fiber Optic Engineers Conference*, pp. 396–401, 2006.

[21] IEEE Standard, *802.1Qay-2009 - IEEE Standard for Local and Metropolitan Area Networks - Virtual Bridged Local Area Networks Amendment 10: Provider Backbone Bridge Traffic Engineering*, 2009.

[22] Internet Engineering Task Force (IETF), *RFC 5921: A Framework for MPLS in Transport Networks.*

[23] D. Fedyk and D. Allan, "Ethernet data plane evolution for provider networks [next-generation Carrier Ethernet transport technologies]," *IEEE Communications Magazine*, vol. 46, pp. 84–89, 2008.

[24] A. Reid, P. Willis, I. Hawkins, and C. Bilton, "Carrier Ethernet," *IEEE Communications Magazine*, vol. 46, pp. 96–103, 2008.

[25] M. Huynh and P. Mohapatra, "Metropolitan Ethernet network: A move from LAN to MAN," *Computer Networks*, vol. 51, pp. 4867–4894, 2007.

[26] S. Salam and A. Sajassi, "Provider Backbone Bridging and MPLS: Complementary technologies for Next Generation Carrier Ethernet transport," *IEEE Communications Magazine*, vol. 46, pp. 77–83, 2008.

[27] S. Vedantham, S. H. Kim, and D. Kataria, "Carrier-grade Ethernet challenges for IPTV deployment," *IEEE Communications Magazine*, vol. 44, pp. 24–31, 2006.

[28] R. Fu, Y. Wang, and M. S. Berger, "Carrier ethernet network control plane based on the Next Generation Network," in *First ITU-T Kaleidoscope Academic Conference*, pp. 293–298, 2008.

[29] M. A. Marsan, A. Bianco, P. Giaccone, E. Leonardi, and F. Neri, "Packet scheduling in input-queued cell-based switches," in *Twentieth Annual Joint Conference on the IEEE Computer and Communications Societies*, 2001.

[30] *High Capacity Carrier Ethernet Transport Networks*, 2010.

[31] K. H. Lee, S. T. Trong, B. G. Lee, and Y. T. Kim, "QoS-guaranteed IPTV service provisioning in IEEE 802.11e WLAN-based home network," in *2008 IEEE Network Operations and Management Symposium Workshops*, pp. 71–76, 2008.

[32] D. Qiu, "On the QoS of IPTV and its effects on home networks," in *5th IEEE Consumer Communications and Networking Conference*, pp. 834–838, 2008.

[33] M. Shreedhar and G. Varghese, "Efficient fair queuing using deficit round-robin," *IEEE/ACM Transactions on Networking*, vol. 4, pp. 375–385, 1996.

[34] International Telecommunication Union (ITU), *G.803 Architecture of transport networks based on the synchronous digital hierarchy (SDH)*, 1997.

[35] C. Wu, H. Wu, and W. Lin, "Delivering relative differentiated services in future high-speed networks using hierarchical dynamic deficit round robin," *Multimeida Systems*, vol. 13, pp. 205–221, 2007.

[36] D. Back, K. Pyun, S. Lee, J. Cho, and N. Kim, "A hierarchical deficit round-robin scheduling algorithm for a high level of fair service," in *2007 International Symposium on Information Technology Convergence*, pp. 115–119, 2007.

[37] S. Jiwasurat, G. Kesidis, and D. Miller, "Hierarchical shaped deficit round-robin scheduling," in *IEEE Global Telecommunications Conference*, pp. 689–693, 2005.

[38] M. Yang, J. Wang, E. Lu, and S. Q. Zheng, "Hierarchical scheduling for diffserv classes," in *IEEE Global Telecommunication Conference*, pp. 707–712, 2004.

[39] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the single-node case," *IEEE/ACM Transactions on Networking*, vol. 1, pp. 344–357, 1993.

[40] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the multiple-node case," *IEEE/ACM Transactions on Networking*, vol. 2, pp. 137–150, 1994.

[41] M. B. Mamoun, J. Fourneau, and N. Pekergin, "Analyzing weighted round robin policies with a stochastic comparison approach," *Computers and Operations Research*, vol. 35, pp. 2420–2431, 2007.

[42] J. C. R. Bennett and H. Zhang, "WF^2Q: worst-case fair weighted fair queueing," in *Fifteenth Annual Joint Conference of the IEEE Computer Socieities (INFOCOM'96)*, pp. 120–128, 1996.

[43] S. J. Golenstani, "A self-clocked fair queueing scheme for broadband applications," in *13th International Conference on Computer Communications (INFOCOM '94)*, pp. 636–646, 1994.

[44] P. Goyal, H. M. Vin, and C. Haichen, "Start-time fair queueing: a scheduling algorithm for integrated services packet switching networks," *IEEE/ACM Transactions on Networking*, vol. 5, pp. 690–704, 1997.

[45] S. S. Kanhere, H. Sethu, and A. B. Parekh, "Fair and efficient packet scheduling using elastic round robin," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, pp. 324–336, 2002.

[46] S. S. Kanhere and H. Sethu, "Fair, efficient and low-latency packet scheduling using nested deficit round robin," in *Workshop on High Performance Switching and Routing*, pp. 6–10, 2011.

[47] D. Saha, S. Mukherjee, and S. Tripathi, "Carry-over round robin: a simple cell shceduling mechanism for ATM networks," *IEEE/ACM Transaction on Networking*, vol. 6, pp. 779–796, 1998.

[48] T. Al-Khasib, H. Alnuweiri, H. Fattah, and V. C. V. Leung, "Fair and efficient frame-based scheduling algorithm for multimedia networks," in *10th IEEE Symposium on Computers and Communications*, pp. 597–603, 2005.

[49] C. Guo, "SRR: An $o(1)$ time-complexity packet scheduler for flows in multiservice packet networks," *IEEE/ACM Transactions on Networking*, vol. 12, pp. 1144–1155, 2004.

[50] C. Guo, "G-3: An $o(1)$ time complexity packet scheduler that provides bounded end-to-end delay," in *2007 IEEE INFOCOM*, pp. 1109–1117, 2007.

[51] C. Guo, "Improved smoothed round robin schedulers for high-speed packet networks," in *2008 IEEE INFOCOM*, pp. 906–914, 2008.

[52] S. Jiwasurat and G. Kesidis, "A class of Shaped Round-Robin (SDRR) schedulers," *Telecommunications Systems*, vol. 25, pp. 173–191, 2004.

[53] A. Varma and D. Stiliadis, "Hardware implementation of fair queuing algorithms for asynchronous transfer mode networks," *IEEE Communications Magazine*, vol. 35, pp. 54–68, 1997.

[54] X. Luo, Y. Jin, Q. Zeng, W. Sun, W. Guo, and W. Hu, "On the stability of multicast flow aggregation in IP over optical network for IPTV delivery," *Chinese Optics Letters*, vol. 6, pp. 553–557, 2008.

[55] Y. J. Won, M. Choi, B. Park, J. W. Hong, H. Lee, C. Hwang, and J. Yoo, "End-user IPTV traffic measurement of residential broadband access networks," in *Network Operations and Management Symposium Workshops 2008*, pp. 95–100, 2008.

[56] OPNET Modeler 16.0, *http://www.opnet.com/*, 2011.

[57] G. A. F. M. Khalaf and S. S. K. El-Yamany, "Statistical multiplexing gain: direct estimation and its application to admission control in ATM networks," in *18th National Radio Science Conference*, pp. 483–496, 2001.

[58] J. Huang, C. W. Tan, M. Chiang, and R. Cendrillon, "Statistical multiplexing over DSL networks," in *26th IEEE International Conference on Computer Communications*, pp. 571–579, 2007.

[59] M. Karol, M. Hluchyj, and S. Morgan, "Input versus output queueing on a space-division packet switch," *IEEE Transcations on Communications*, vol. 35, pp. 1347–1356, 1987.

[60] D. Pan and Y. Yang, "Fifo-based multicast scheduling algorithm for virtual output queued packet switches," *IEEE Transaction on Computers*, vol. 54, pp. 1283–1297, 2005.

[61] D. Pan and Y. Yang, "Bandwidth guaranteed multicast scheulding for virtual output queued packet switches," *Journal of Parallel and Distributed Computing*, vol. 69, pp. 939–949, 2009.

[62] B. Prabhakar, N. McKweon, and R. Ahuja, "Multicast scheduling for input-queued switches," *IEEE Journal on Selected Areas in Communications*, vol. 15, pp. 855–866, 1997.

[63] A. Bianco, P. Giaccone, C. Piglione, and S. Sessa, "Practical algorithms for multicast support in input queued switches," in *IEEE International Conference on High Performance Switching and Routing*, pp. 187–192, 2006.

[64] A. Mekkittikul and N. McKeown, "A practical scheduling algorithm to achieve 100% throughput in input-queued switches," in *17th Annual Joint Conference of the IEEE Computer and Communications Societies*, pp. 792–799, 1998.

[65] H. J. Chao, "Next generation routers," *Proceedings of the IEEE*, vol. 90, pp. 1518–1558, 2002.

[66] S. Gupta and A. Aziz, "Multicast scheduling for switches with multiple input-queues," in *10th Symposium on High Performance Interconnects*, pp. 28–33, 2002.

[67] M. Shoaib, "Selectively weighted multicast scheduling designs for input-queued switches," in *2007 IEEE International Symposium on Signal Processing and Information Technology*, pp. 92–97, 2007.

[68] L. Mhamdia and S. Vassiliadis, "Integrating uni- and multicast scheudling in buffered crossbar switches," in *IEEE International Conference on High Performance Switching and Routing*, 2006.

[69] Cisco Product Overview, http://www.cisco.com, *Cicso 12000 Gigabit Switch Router*, March 2011.

[70] N. McKeown, M. Izzard, B. E. A. Mekkittikul, and M. Horowitz, "The tiny tera: a packet switch core," *IEEE Micro Magazine*, vol. 17, pp. 27–40, 1997.

[71] H. Duan, J. W. Lockwood, S. M. Kang, and J. D. Will, "A high performance oc-12/oc-48 queue design prototyp for input bufferec atm switches," in *Sixteenth Annual Joint Conference on the IEEE Computer and Communications Societies*, vol. 1.

[72] T. Anderson, S. Owick, J. Saxe, and C. Thacher, "High-speed switch scheduling for local-area networks," *IEEE/ACM Transcations on Networking*, vol. 11, pp. 319–352, 1993.

[73] Y. Tamir and G. Frazier, "High performance multi-queue buffers for vlsi communication switches," in *15th Annual International Symposium on Computer Architecture*, pp. 343–354, 1988.

[74] N. McKeown, "The iSLIP scheduling algorithm for input-queued switches," *IEEE/ACM Transcations on Networking*, vol. 7, pp. 188–201, 1999.

[75] M. A. Marsan, A. Bianco, P. Giaccone, E. Leonardi, and F. Neri, "Multicast traffic in input-queued switches: optimal scheduling and maximum throughput," *IEEE/ACM Transations on Networking*, vol. 11, pp. 465–477, 2003.

[76] J. Hayes, R. Breault, and M. Mehmet-Ali, "Performance analysis of a multicast switch," *IEEE Transcations on Communications*, vol. 39, pp. 581–587, 1991.

[77] X. Li, Z. Zhou, and M. Hamdi, "Space-memory-memory architecture for Clos-network packet switches," in *IEEE International Conference on Communications*, pp. 1031–1035, 2005.

[78] S. Sun, S. He, Y. Zheng, and W. Gao, "Multicast scheduling in buffered crossbar switches with multiple input queues," in *IEEE International Conference on High Performance Switching and Routing*, pp. 73–77, 2005.

[79] F. Abel, C. Minkenberg, I. Iliadis, T. Engbersen, M. Gusat, F. Gramsamer, and R. P. Luijten, "Design issues in next-generation merchant switch fabrics," *IEEE/ACM Transactions on Networking*, vol. 15, pp. 1603–1615, 2007.

[80] K. Pun and M. Hamdi, "Distro: A distributed static round-robin scheduling algorithm for bufferless Clos-network switches," in *IEEE Global Communications Conference*, pp. 2298–2302, 2002.

[81] R. Rojas-Cessa, E. Oki, and J. Chao, "Maximum weight matching dispatching scheme in buffered Clos-network packet switches," in *IEEE International Conference on Communications*, pp. 1075–1079, 2004.

[82] E. Oki, Z. Jing, R. Rojas-Cessa, and J. Chao, "Concurrent round-robin-based dispatching schemes for Clos-network switches," *IEEE/ACM Transactions on Networking*, vol. 10, pp. 830–844, 2002.

[83] Y. Yang and G. M. Masson, "The necessary conditions for Clos-type nonblocking multicast networks," *IEEE Transactions on Computers*, vol. 48, pp. 1214–1227, 1999.

[84] Y. Yang and J. Wang, "On blocking probability of multicast networks," *IEEE Transactions on Communications*, vol. 46, pp. 957–968, 1998.

List of Acronyms

CAC	Call Admission Control
CD	Cell Dispatching
CM	Central Module
CMF	Credit based Multicast Fair
CMSD	Concurrent Master-Slave round-robin Dispatching
CORR	Carry-Over Round Robin
CRRD	Concurrent Round-Robin Dispatching
DC	Deficit Counter
DRR	Deficit Round Robin
DSLAM	Digital Subscriber Line Access Multiplexer
DSRR	Desynchronized Static Round Robin
ERR	Elastic Round Robin
FIFO	First-In-First-Out
FIFOMS	FIFO-based Multicast Scheduling
GMSS	Greedy Min-Split Scheduling
GPS	Generalized Processor Sharing
HD	High Definition

HOL	Head-Of-Line
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IM	Input Module
IP	Internet Protocol
IPP	Input Port Processor
IPTV	Internet Protocol Television
IQ	Input Queuing
ITU-T	International Telecommunication Union-Telecommunication Standardization Sector
L1	Layer 1
L2	Layer 2
L3	Layer 3
LA	Look-Ahead
LAN	Local Area Network
MAC	Media Access Control
MAN	Metropolitan Area Network
MC-VOQ	MultiCast Virtual Output Queuing
MEF	Metro Ethernet Forum
MF-DSRR	Multicast Flow-based DSRR
MFRR	Multicast Flow-based Round-Robin
MLRRMS	Multi-Level Round-Robin Multicast Scheduling
MMM	Memory-Memory-Memory

MPLS	Multi-protocol Label Switching
MPLS-TP	Multi-protocol Label Switching Transport Profile
MRR	Mini Round Robin
MSM	Memory-Space-Memory
MWMD	Maximum Weight Matching Dispatching
NGN	Next Generation Network
OAM	Operation, Administration and Maintenance
OM	Output Module
OOS	Out-Of-Sequence
OPP	Output Port Processor
OQ	Output Queuing
PBB-TE	Provider Backbone Bridge with Traffic Engineering
PSTN	Public Switched Telephone Network
PW	Packet Weight
QoS	Quality-of-Service
SCFQ	Self-Clocked Fair Queuing
SDH	Synchronous Digital Hierarchy
SE	Switching Element
SFQ	Start-time Fair Queuing
SMG	Statistical Multiplexing Gain
SMM	Space-Memory-Memory
SONET	Synchronous Optical Networking
SRRD	Static Round-Robin Dispatching

S^3	Space-Space-Space
STB	Set Top Box
T-MPLS	Transport MPLS
VLAN	Virtual Local Area Network
VoD	Video-on-Demand
VoIP	Voice-over-IP
VOQ	Virtual Output Queuing
WBA	Weight-Based Algorithm
WFQ	Weighted Fair Queuing
WF^2Q	Worst-case Fair Weighted Fair Queuing